The Use of Alternative Data Models in Data Warehousing Environments

Victor González Castro

Submitted for the degree of Doctor of Philosophy

Heriot-Watt University

School of Mathematical and Computer Sciences

Edinburgh, United Kingdom

May 2009

Abstract

Data Warehouses are increasing their data volume at an accelerated rate; high disk space consumption; slow query response time and complex database administration are common problems in these environments. The lack of a proper data model and an adequate architecture specifically targeted towards these environments are the root causes of these problems.

Inefficient management of stored data includes duplicate values at column level and poor management of data sparsity which derives from a low data density, and affects the final size of Data Warehouses. It has been demonstrated that the Relational Model and Relational technology are not the best techniques for managing duplicates and data sparsity.

The novelty of this research is to compare some data models considering their data density and their data sparsity management to optimise Data Warehouse environments. The Binary-Relational, the Associative/Triple Store and the Transrelational models have been investigated and based on the research results a novel Alternative Data Warehouse Reference architectural configuration has been defined.

For the Transrelational model, no database implementation existed. Therefore it was necessary to develop an instantiation of it's storage mechanism, and as far as could be determined this is the first public domain instantiation available of the storage mechanism for the Transrelational model.

Dedication

This thesis is dedicated to my beloved girls Pilar, Pily and Vicky

Acknowledgements

It has been a great pleasure to undertake this PhD at Heriot-Watt University. It has been highly demanding and of course it has not been easy. I want to thank the many people who have been my support during these years.

First of all I want to thank my loved daughters Pilar and Victoria because they have supported very well during this adventure in a foreign country - our now beloved Scotland. When my dear wife Pilar and I decided to study our PhD and came to the UK it was our decision but for our poor girls it was an imposition. It has not been easy for both of you Pily and Vicky but thank you very much for being as good daughters as you are, and I hope you will never forget Edinburgh and your English language.

Of course I want to thank in a very special way to my soul mate Pilar, without you my beloved wife, I would not have any motivation in this world; you are my inspiration and my encouragement to do everything I do. If you were not as adventurous as you are, we might have never started and completed our PhD studies; we were so fine in Mexico and then you came with this crazy and wonderful idea to study abroad.

Thanks to my three girls Pilar, Pily and Vicky for all those years together, all those afternoons and weekends of study, and those days of boring museums and statues.

To my supervisors and friends, Professor: Lachlan MacKinnon thank you for all your valuable advice and for share your knowledge with me. I really appreciate your help. Thank you for those days when I required your supervision and when I asked you to review my documents in a hurry. David Marwick, thanks for accepting me as student, you have really helped me when I needed it, and thank you for believing in me.

My thanks go to Greg Michelson and to all the staff of Heriot-Watt University who really supported me during these years. Thank you to all my PhD peers Alasdair, Elizabeth, Ke, Lisha, Mike, Alexander and the rest of you.

Thank you to all the Mexicans in Edinburgh: you are my extended family here in Scotland. Thank you very much for all those Fridays of playing table games and sometimes drinking; none of you never required anything special: it was enough to be a Mexican and you opened your hearts to all of us. Especially Caro and Sam.

Of course I want to thank my country –México- for supporting me via the Consejo Nacional de Ciencia y Tecnología (CONACYT) with the scholarship granted; also I want to thank the complementary scholarship granted via the Dirección General de Relaciones Internacionales de la Secretaria de Educación Pública (SEP); without their supports I could not have studied y my PhD.

My thanks go to my father, brothers and sisters for their support and in memory of my mother.

ACADEMIC REGISTRY
Research Thesis Submission

Name*:*	Victor González Castro		
School/PGI:	School of Mathematical and Computer Sciences		
Version: *(i.e. First, Resubmission, Final)*	Final	Degree Sought (Award **and** Subject area)	PhD Computer Science

Declaration

In accordance with the appropriate regulations I hereby submit my thesis and I declare that:

1) the thesis embodies the results of my own work and has been composed by myself
2) where appropriate, I have made acknowledgement of the work of others and have made reference to work carried out in collaboration with other persons
3) the thesis is the correct version of the thesis for submission and is the same version as any electronic versions submitted*.
4) my thesis for the award referred to, deposited in the Heriot-Watt University Library, should be made available for loan or photocopying and be available via the Institutional Repository, subject to such conditions as the Librarian may require
5) I understand that as a student of the University I am required to abide by the Regulations of the University and to conform to its discipline.

* *Please note that it is the responsibility of the candidate to ensure that the correct version of the thesis is submitted.*

Signature of Candidate*:*		Date:	May-2009

Submission

Submitted By *(name in capitals):*	VICTOR GONZALEZ CASTRO
Signature of Individual Submitting:	
Date Submitted:	May-2009

For Completion in Academic Registry

Received in the Academic Registry by *(name in capitals):*			
Method of Submission *(Handed in to Academic Registry; posted through internal/external mail):*			
*E-thesis Submitted (**mandatory for final theses from January 2009)***			
Signature:		Date:	

Table of Contents

List of Tables

List of Figures

List of Publications

Some of the material presented in this thesis has already appeared in the following conference papers and a journal.

Conference papers:

- Gonzalez-Castro, Victor. MacKinnon, Lachlan. A Survey "Off the Record" – Using Alternative Data Models to Increase Data Density in Data Warehouse Environments. Proceedings of the 21st British National Conference on Databases BNCOD Volume 2. Edinburgh, Scotland 2004.
- Gonzalez-Castro, Victor. MacKinnon, Lachlan. Using Alternative Data Models to increase data density in Data Warehouse environments. Proceedings of the Postgraduate Research Conference in Electronics, Photonics, Communications and Networks, and Computing Science - PREP 2005. pp. 128-129. Lancaster, England March 2005.
- Gonzalez-Castro, Victor. MacKinnon, Lachlan. Data Density of Alternative Data Models and its Benefits in Data Warehousing Environments. British National Conference on Databases BNCOD 22 Proceedings Volume 2. pp 21-24. Sunderland, England U.K. 2005.
- Petratos, P. and Michalopoulos, D. (editors). Gonzalez-Castro, V. and MacKinnon, L. (authors). Using Alternative Data Models in the context of Data Warehousing. 1st International Conference on Computer Science and Information Systems by the Athens Institute of Education and Research ATINER. Athens, Greece, 2005. pp. 83-100. ISBN-960-88672-3-1.
- Gonzalez-Castro, Victor. MacKinnon, Lachlan. Marwick, David. An Experimental Consideration of the Use of the Transrelational Model for Data Warehousing. Proceedings of the 23rd British National Conference on Databases BNCOD. Belfast, Northern Ireland, 2006.

Journal paper:

- Olivares Ceja, Jesus. Guzman Arenas, Adolfo. (editors). Gonzalez-Castro, Victor. MacKinnon, Lachlan. Marwick, David. (authors) A Performance Evaluation of Vertical and Horizontal Data Models in Data Warehousing. Research in Computing Science Volume 22 November 2006. Special issue: Data Mining and Information Systems. Instituto Politécnico Nacional, Centro de Investigación en Computación, México. pp. 67-78 ISSN: 1870-4069

Technical Report:

- Gonzalez-Castro, Victor. MacKinnon, Lachlan. A Technical Report “Off the Record”: Using Alternative Data Models to increase Data Density in Data Warehouse Environments. Technical Report HW-MACS-TR-0024. Heriot-Watt University Technical Report Series. 1st of November, 2004.

The following papers have been accepted in the 2009 British National Conference on Databases and will be published by July 2009.

- Gonzalez-Castro, Victor. MacKinnon, Lachlan, Angeles Maria del Pilar. An alternative Data Warehouse Architectural Configuration. British National Conference on Databases BNCOD 26. July 2009. Birmingham, England U.K. 2005.
- Gonzalez-Castro, Victor. MacKinnon, Lachlan, Angeles Maria del Pilar. The Use of the Binary-Relational Model in Industry, a Practical Approach. British National Conference on Databases BNCOD 26. July 2009. Birmingham, England U.K.

CHAPTER 1
Introduction

1.1 Background

Decision Support Systems (DSS) have existed for many years [Gill, 1996]. DSS have been based on different technologies, from 3GL fix reports to the dynamic On Line Analytical Processing (OLAP) tools, or even more sophisticated Data mining tools. All those systems are based on a Data Warehouse backend, which stores historical information that allows such systems to support decisions based on factual information.

As the benefits of DSS systems are tangible, for example the retention of existing customers or attracting new customers [Freeman, 1997] by making a discounting promotion on an specific city, organizations have been quick to implement Data Warehouses. The kind of decision support tasks considered in this thesis is the kind of decision where the end-user interacts with the system, more like the OLAP kind of analysis than the data mining approach, where the system runs a set of analyses to find relations between data, but the response time could not be a major issue. In contrast, in OLAP analysis, the end-user is in front of a computer waiting for the system to answer the query; therefore if the DBMS is slow then the end-user can lose interest in the business analysis, leave the query running and forget the business question they wanted originally. In order to support more decisions through the entire organization, more data are stored in the Data Warehouses. Consequently, data repositories have grown at an exponential rate [Freeman, 1997]. Business decisions are taken and supported based on historical data stored in Data Warehouses.

On the other hand, huge amounts of data cause problems for the Information Technology (IT) people who struggle to get the right information at the right time. The problems faced by IT organizations are the motivation behind this thesis and are listed in section 1.4.1. Those problems are caused by the lack of an adequate data model for Data Warehousing, which can work harmoniously within an Alternative Data Warehouse (ADW) Architectural Configuration. ADW is fully described in Chapter 8. It is referred to as Alternative Architectural Configuration because it does not use the traditional n-ary-Relational model approach (also referred to as the Relational Model).

As the Relational technology has been successful in transactional systems or OLTP (On Line Transactional Processing), its use in Data Warehousing has been extrapolated without question.

1.2 Data Density and Data Sparsity

The initial purpose of this research was to find a way to reduce the final Data Warehouse size as this is the initial manifestation of the problem that industry is facing. Data Density is defined as the amount of useful information per storage unit (i.e. rows per disk MB). Data Sparsity is defined as the ratio between existing and non-existing information per storage unit. It is a fact that a reduction in Data Sparsity means an increase on Data Density. One way to increase Data Density and reduce Data Sparsity is the use of compression algorithms. However the use of compression algorithms is not enough (see section 2.5 of this thesis) as they can increase data density, but their use on RDBMs is unfeasible because of the processing time required to compress and uncompress data [Westmann, 2000]. Compression is a technology-based approach but the fundamental reason is not addressed. Therefore, an investigation of alternative data models which can better handle data sparsity and achieve better data density was carried out.

It was found (Chapter 3) that the use of an non n-ary-Relational data model (referred to as an alternative data model) satisfies both issues: it is able to reduce data sparsity and increase data density, and using this approach the Data Warehouse growth problem is addressed from the fundamental reason (the data model behind the DBMS) and not only with technology-based solutions (compression and other techniques). Once these two issues have been properly addressed, it is possible to move forward and define an Alternative Data Warehouse Reference Architecture based on a non n-ary-Relational model.

1.3 The ADW Reference Architectural Configuration

The final purpose of this thesis is to define an Alternative Data Warehouse (ADW) reference architectural configuration that would be specifically tailored towards Data Warehouse environments. Such architectural configuration addresses the main problems faced by current n-ary-Relational Data Warehouses as database explosion, data sparsity, low data density, long batch processing and low query performance. The ADW reference architectural configuration described in Chapter 8 was conceived after analysing and benchmarking various data models which abandon the record storage structure. ADW is based on the research work carried out during the PhD, the results of which are included in the remainder of this thesis.

1.4 Research Goals and Contributions

In this section the research goals and contributions are defined. These were the main directors of the research carried out during this thesis.

1.4.1 Motivations for Research

Experience on building and maintaining Data Warehouses for a number of different companies [Freeman, 1997] has consistently demonstrated that Data Warehouses are growing at an exponential rate. Therefore Data Warehouses contain Terabyte-sized tables.

The factors that contribute to the Data Warehouse exponential growth are:

1. Depth of history maintained
2. Level of data granularity
3. Number of dimensions involved
4. Dimension levels
5. Pre-calculation of all possible aggregate level query answers
6. Data completeness
7. Data sparsity
8. Ability of the Database Management System (DBMS) to handle data sparsity
9. DBMS capacity to handle and recover space in fragmented disk table spaces
10. DBMS data compression features

In order to control the above factors, partial solutions have been adopted, such as:

1. *Depth of history maintained*. Depth of history depends on the business needs for maintaining historical data. Organizations limit the number of years kept on-line in order to deal with storage and processing limitations [Kimball, 1996]; however they lose part of Data Warehouse richness by doing so.

2. *Level of data granularity*. Commonly organizations follow the approach to keep 13 months [Freemann, 1997] of data at the lowest level. Such a level of granularity allows the execution of a *this-year-last-year analysis*, and after this period of time, data are summarized to higher levels of granularity [Kimball, 1996].
3. *Number of dimensions involved*. Dimensions are used to represent the different points of view of the organization's data. It is common to limit the Data Warehouse design to a particular business area by creating Data Marts instead of a real Enterprise Data Warehouse [Kimball, 1996].
4. *Dimension levels*. The decision made during the conceptual design of a Data Warehouse to limit the number of dimension levels (Hierarchy depth) to the most common levels of analysis (for example aggregates to a country or region instead of performing analysis at the shop level [Kimball, 1996]).
5. *Pre-calculation of all possible aggregate level query answers*. The balance between the desired online answer time and the batch pre-calculation time to complete all possible combinations should be made. This trade-off is considered in order to determine which possible aggregated queries will be pre-computed. Several techniques have been proposed (approximate queries [Acharya, 1999], iceberg cube calculation [Beyer, 1999], aggregate materialized views [Agrawal, 1997a] and summary tables [Datta, 2004]) to mitigate this problem. If this problem is not properly managed, it will lead to the *Database explosion* phenomenon [Pendse, 2003b].
6. *Data Completeness*. During the design of a Data Warehouse end users want some analyses that they regard as interesting, but the data required to build these analyses do not exist within the working databases. Sometimes users have to forget about such analyses. However, if users regard these requirements as paramount, considerable additional data may have to be generated to mitigate this situation.
7. *Data sparsity*. Defined as non-existent information, this is one of the issues that has been investigated within Data Warehouse environments but is still unresolved. If there are "holes" in the data, then creating information to fill it is not trivial. Unfortunately the data simply do not exist and to fill them with any value is meaningless. In Relational technology, and particularly in SQL, the management of

missing information has been a significant area of research, but it has been more oriented to OLTP systems and it has not been properly solved [Codd, 1990].

8. *Ability of the Database Management System to handle data sparsity*. Commercial RDBMS are built on SQL capacities and limitations [Date, 1987] but in practice commercial RDBMS do not handle data sparsity well [Codd, 1990]. This is also true in Multidimensional Databases [Li, 1999].
9. *DBMS capacity to handle and recover space in fragmented disk table spaces*. This is handled in different ways by the DBMS. For example the ADABAS-D RDBMS is constantly moving records within the data space minimizing data fragmentation and constantly recovering fragmented disk space [SoftwareAG, 1997a], [SoftwareAG, 1997b], but those that have a deficient space recovery mechanism are not well suited for Data Warehouse environments. For those DBMSs which require unload and reload tables to recover the space, this is not a practical solution for Terabyte-sized tables, because of the temporary space required for the operation as well as the processing time.
10. *DBMS data compression.* RDBMS avoid the use of traditional compression methods such a Huffman coding and Lempel-Ziv to reduce the size of the Database. This is principally because of the CPU cost involved in undertaking compression and decompression operations [O'Connel, 2003].

Some people consider that disk memory is cheap. Therefore, the pre-calculation of all possible aggregation levels and combinations are performed without considering the Data Warehouse growth. Special attention should be taken because an unexpected behaviour of multidimensional data can occur. Usually the input data is very sparse, which means that the majority of possible cells defined as combinations of the dimension levels actually do not contain data. Distributed input data values can have hundreds of computed dependent cells. The computed space is much denser than the input data, and the result is that pre-computed results based on sparse data are more voluminous than might be expected [Pendse, 2003b]. The original data set can grow by factors of tens or even hundreds [Pendse, 2003b]. When considering Terabyte sized Data Warehouses, the storage cost is non-trivial. Data storage maintenance costs are often seven times as high per year as the initial purchase cost [Datta, 2004; Simpson, 1997]. If the initial media cost is for example $100,000 USD, then the additional cost would be $700,000 USD for every year it is operational [Simpson, 1997]. In conclusion, the storage size and its maintenance are important issues to be managed.

Examples of associated costs that are sometimes forgotten are: computer centre space, disks' electricity consumption, air conditioning for cooling the disks, disk repairs, tape drivers and tape media used to backup disk data.

There is a direct relationship between Data Warehouse growth and the growth of the query response time and the maintenance task-processing times. Commonly organizations find that undertaking Data Warehouse maintenance overnight is not enough, and some of these tasks need to be minimized, alternated or even discarded (i.e. incremental backups vs. full backups, backups every third day instead of daily, partial statistics computation vs. full statistics, etc.) Such decisions increase the risk of losing information within the Data Warehouse environment.

Partial remedies, as previously mentioned, are the ones used today. However, the fundamental reason has not been addressed, which is the lack of an Architecture based on a data model better suited for Data Warehouse environments.

The Relational model is the predominant model used in commercial RDBMS which have been demonstrated to be very successful in transactional environments. However, RDBMS have also been used to create Data Warehouses without questioning their suitability; but in this research the following characteristics of Relational Data Warehouses have been noticed:

1. Data Warehouses grow at an exponential rate [Datta, 2004].
2. The Database explosion phenomenon [Pendse, 2003b] is difficult to control or eliminate.
3. Poor management of data sparsity [Gonzalez, 2004].
4. Low Data Density [Gonzalez, 2005a].
5. Huge amounts of disk storage are required [Zukowski, 2005].
6. The cost of storage and its maintenance are not negligible [Datta, 2004] and the storage itself could be up to 80% of the entire system cost [Zukowski, 2005].
7. High query processing times is one of the main user complaints. According to [Pendse, 2003a], 17% of all sites using OLAP tools complained about the query

response time, with the worst case reaching 42% of the sites that use Oracle Discoverer.

8. Long periods of time to Extract, Transform and Load (ETL) data [Gonzalez, 2006b].
9. Big batch processing windows to backup and restore the environment [Gonzalez, 2006b].
10. High complexity of the Database Administration tasks Including Index creation and its maintenance, and long times required to compute statistics [Gonzalez, 2006b].

The above-mentioned problems are the motivators for this research, the results of which are presented in this thesis.

1.4.2 Research Hypothesis

Some non n-ary-relational models (referred to as alternative data models) have been investigated (Chapter 3) in order to find a data model that could be useful in Data Warehousing environments. Some alternative data models follow a vertical storage approach, and are the ones considered in this research. These models abandon the traditional horizontal record storage structure (n-ary storage systems [Date, 2004]). Isolated studies have been made for some of the models considered [MonetDB, 2004a] [Williams, 2003; Date, 2004]. None of these studies has considered as broad an approach as the one taken on this thesis (Chapter 3, Chapter 6 and Chapter 7). This research considers specifically the use of alternative data models within the context of Data Warehousing, not for any other kind of environment (Chapter 3, Chapter 4, Chapter 6 and Chapter 7). Apart from that, the current study has been conducted impartially without trying to favour any particular data model or technology, in contrast with other studies that were biased towards a particular data model or technology (Chapter 4).

The rationale behind the use of those models, which follow a vertical approach, is that duplicate values at a column level are eliminated - including nulls. Such an approach increases data density (rows stored per MB on disk) and reduces data sparsity on the Data Warehouse. The hypothesis on which this research was based is as follows:

"Data density in data warehouse environments can be improved by using alternative data models"

1.4.3 Research Goals and Thesis Objectives

Objective 1: Identify an alternative data model, which increases data density.

Objective 2: Find an alternative data model, which better handles data sparsity than the n-ary-relational model.

Objective 3: Demonstrate that the selected model better handles data sparsity and allows the improvements of data density in Data Warehouse environments by benchmarking the alternative model against the relational model.

Objective 4: Create a novel Alternative Data Warehouse (ADW) reference architectural configuration, which uses the data model best suited for Data Warehouse environments.

Objective 5: ADW should maintain compatibility with the widely deployed relational technology.

Objective 6: By using the alternative data model and the reference architectural configuration, help to solve the previously mentioned storage, querying and maintenance challenges in Data Warehousing.

1.4.4 Contributions to research

The contributions to research of this thesis are:

Contribution 1: Some of the alternative data models have existed for many years in the computer science body of knowledge, but they have been studied for transactional environments, not for Data Warehouse environments as has been done in this research (Chapter 3, Chapter 6 and Chapter 7 of this thesis).

Contribution 2: Extensions to the TPC-H standard benchmark [TPCH, 2002] have been made in order to consider the full operation cycle of a production Data Warehouse environment (i.e. Backup and Restore times), plus some additional metrics that must be considered, such as the Data Density of a Data Warehouse (Chapter 4).

Contribution 3: The first public domain implementation of the storage mechanism for the Transrelational^{TM} model has been made during this research (Chapter 5) and has been reported in [Gonzalez, 2006a].

Contribution 4: A novel Alternative Data Warehouse (ADW) reference architectural configuration has been developed. ADW is based on the alternative data model best suited for Data Warehousing environments (Chapter 8).

Contribution 5: An explicit classification of existing cube approaches has been made in section 2.4.4. Many cube approaches exist in the research body of knowledge, and as far as is known no-one has made a clear classification of these cube approaches in the past.

Contribution 6: Improvements to the Transrelational model algorithms have been made on Chapter 5. In particular the *Column Condensation Threshold* concept has been introduced to the Date's proposed algorithm [Date, 2004] to compute the Field Values Table used by the Transrelational model (refer to section 5.4.1) Also *an alternative algorithm* to compute the Record Reconstruction Table has been proposed (refer to section 5.4.2).

1.5 Structure of Thesis

This thesis is laid out in nine chapters. This introductory chapter provides an initial description of the rationale for the work undertaken, the problems that would be addressed and the Alternative Data Warehouse reference architectural configuration that has been developed.

Chapter 2 is a background chapter, which considers the current n-ary-relational Data Warehousing Environments and their characteristics. It also introduces some common terminology used in Data Warehousing, as well as summarizing some techniques used to increase performance in Relational Data Warehouses; and finally it presents a traditional Relational Data Warehouse Architecture.

Chapter 3 considers various alternative data models that abandon the n-ary horizontal storage approach. The storage structures of the Relational model [Codd, 1990] are described. This is important because it will be the base model against which all other models will be compared. Then the Associative/Triple Store model [Williams, 2003; Sharman, 1988] is analysed followed by the Binary-Relational [Copeland, 1985] and the Transrelational^{TM} [Date, 2004] models. Those are the alternative data models that were chosen, analysed and benchmarked.

Chapter 4 presents the Benchmarking Standards that were studied and the reasons for choosing the TPC-H [TPCH, 2002] benchmark. It was chosen as it is a well-accepted benchmark in both research and industry. This chapter also presents the extensions made to the TPC-H. It was necessary to extend the TPC-H benchmark in order to consider the whole operation cycle followed on Data Warehouses.

Chapter 5 presents an implementation of the storage mechanism for the Transrelational Model and the results of its comparison against the other models. Such implementation is novel and as far as could be determined it is the first available public domain implementation of the storage mechanism for the Transrelational^{TM} model.

Chapter 6 this chapter presents the benchmarking of the models. A set of metrics has been evaluated for each data model in order to find the best-suited data model for Data Warehousing environments.

Chapter 7 this chapter analyses the results obtained in Chapter 6 and determines and presents the best alternative data model for Data Warehousing.

Chapter 8 presents the Alternative Data Warehouse (ADW) reference architectural configuration which considers the best suited alternative data model chosen for Data Warehousing Environments. The ADW is proposed as a reference architectural configuration to be followed for future and existing Data Warehouses because compatibility with the n-ary-Relational installed base has been considered.

Chapter 9 provides a summary of the major points from each chapter and the conclusions of this thesis, and presents the novelty of the approach followed by applicability of the ADW reference architectural configuration and the future directions of this research area.

This thesis contains appendices, where some sample programs are presented; these programs allow further comparisons between the models studied.

CHAPTER 2
Background

2.1 The Relational Model

The Relational Model is today's dominant paradigm. Therefore, in this thesis, a comparison of alternative data models against the relational model has been made. The well-known paper by Codd "A Relational Model of Data for Large Shared Data Banks" [Codd, 1970] introduced the mathematical foundation for the relational model for the first time.

2.1.1 Relational Model and Relational Technology Strengths

The strengths of the relational model and the Relational technology are:

- It is the most accepted data model and technology. It is widely deployed in industry [Codd, 1990].
- Even though its implementation has not been good [Codd, 1990], it is the best approach available to manage databases.
- It has a solid mathematical foundation [Codd, 1990].
- People know how to manage relational technology but without understanding the relational model [Date, 2005].

Unfortunately, its commercial success blinds people from researching the use of other data models in Data Warehousing environments. People have simply extrapolated the use of relational products because they feel comfortable about managing such products.

2.1.2 Relational Model/Technology Weaknesses

- The Relational Model has not been fully implemented according to its mathematical model [Codd, 1990].
- The Relational Model, Relational Technology, and the SQL language concepts are commonly confused [Date, 2005].
- Relational implementations follow an n-ary horizontal data representation storing records [Date, 2004]. This is not the best approach for Data Warehouse

environments because typical analytical queries retrieve only few attributes within the tuples [Teklitz, 2003]; In the case of Data Warehouses, a vertical approach to data representation is better because only the attributes involved in the query are retrieved and the access of attributes which are not required to answer the query is avoided [Teklitz, 2003].

Chen, in his fundamental paper "The Entity-Relationship Model –Toward a unified view of data" [Chen, 1976], gives a graphical representation of how to model a database, which facilitates the understanding of database logical modelling. Entity Relationship modelling is the most popular and fundamental technique in current database modelling. Such is the acceptance of this modelling technique that it has also been extrapolated as well to modelling Data Warehouses. Examples of such modelling techniques are Star and Snowflake logical data modelling, which are common in the Data Warehouse arena [Kimball, 1996].

These modelling techniques are analysed in section 2.3.1. As these are the most used Data Warehousing modelling techniques, they must be supported at the logical level by the alternative data model chosen.

2.2 The SQL Language, Strengths, Weaknesses and Industrial acceptance

The SQL language is poor and weak as a programming language [Date, 1987] and specifically as relational language. In Darwen [1995], a full compliance Relational language is defined as "D" language, but unfortunately, all commercial products use SQL [Codd, 1990; Date, 2005]. The classical paper by Date, "Where SQL falls short" [Date, 1987], provides the basis from which to understand such deficiencies.

"Orthogonal programming language design means that distinct language concepts are clearly separated, not bundled together. The advantage of orthogonal language design is that it leads to a language that is coherent – one that possesses a simple, clean, and consistent structure, in both its syntax and semantics. There are no exceptions, special cases, or unpleasant surprises for the user. Unfortunately, SQL is not orthogonal. It is full of apparently arbitrary restrictions, exceptions, and special rules. Ironically, restrictions have the effect of simultaneously increasing the size of the language while decreasing its power or functionality" [Date, 1987]. *"Orthogonality has been a well-established language design for many years before SQL was defined. There was no*

justification for ignoring orthogonal design when SQL was implemented as commercial product in the 80´s" [Date, 1987].

SQL omits important relational features such as duplicate rows. SQL provides a very informal handling of NULLS [Codd, 1990], making its representation suitable for errors. Furthermore, SQL accepts duplicate values, which is a mistake since mathematical relations are unique by definition [Codd, 1990]. From here, it is important to highlight the weaknesses of SQL as two of its main deficiencies are directly related to the current study. These are:

- The deficient management of Null values. These are related to the data sparsity problem (objective 2 of the thesis) currently present in all the relational Data Warehouses.
- The duplicate values allowed by SQL. When the data density problem was addressed (research objective 1), the acceptance of duplicate values within the Database had the effect of creating a low data density measurement.

When looking for an alternative data model, data density was one of the main features to consider. If an alternative data model can store just one value and use it as many times as required, then it will have the effect of increasing the Data Density. If the alternative data model is able to store just one representation of Null and used as many times as required, it increases the data density and reduces the data sparsity even if it is accessed by an SQL language interface.

In 1990, Codd published the formal compendium of the Relational Model [Codd, 1990] and he called it *The relational Model Version 2.* Here Codd tried to solve a number of mistakes made by vendors during the implementation of Version 1, such as the following:

a. Duplicate rows permitted by the language SQL.
b. Primary keys either omitted altogether, or they have been made optional on base relations.
c. Major omissions, especially of all features supporting the meaning of the data (including domains).
d. Indexes misused to support semantic aspects.

e. Omission of almost all the features concerned with preserving the integrity of the database.

A major fundamental change in the relational model version 2 is introduced. Codd accepted the weakness of the relational model in handling nulls [Codd, 1990] by using a Three-valued logic. In the second version of the Relational Model, the introduction of a Four-valued logic is made in order to improve the management of null values. There are two kinds of missing information:

- Missing but applicable, denoted by “A-mark”. It is when a particular property is applicable to the particular object, but at the current moment, it is not available. For example, an employee has a missing but applicable present salary, but we do not know at the moment what it is.
- Missing but inapplicable denoted by “I-Mark”. It is when a particular property is inapplicable to the particular object represented by the row, for example, commission on non-sales personnel.

Unfortunately, there is a not a full implementation of the Relational Model in either Version 1 or Version 2 [Codd, 1990].

The study of missing information is important for the current study, because missing information is an essential part of data sparsity, which is the research objective 2. It relates to the data density (objective 1), and its final effect is the growth of the Data Warehouse.

2.3 Data warehousing Background

During the years the automation of business process through computerized systems has been growing, consequently the amount of data keep on the systems has been growing at accelerated rate; however the systems have been designed isolated from each other with little or no integration and each system is managing its own data set. As a result the same data is represented and stored in many different ways, one for each system; consequently there are multiple versions of the truth. The main objective of a Data Warehouse is to be the unique source of the truth for the companies, and here comes the challenges for Data Warehousing: Integrate data, consolidate data, cleansing the data, store historical data and provide answers to business questions in a faster way.

According to [Chaudhuri, 1997], “Data Warehousing is a collection of decision support technologies that aim at enabling an enterprise to make better and faster decisions. Finding the information at the right time is necessary for companies to make the proper decisions”.

The Data Warehouse origin can be traced down to the research carried out at the MIT in the 1970s [Haisten, 2003]. Here for the first time a differentiation between the operational and analytical processing is made. Later in 1988, Devlin and Murphy introduced the term “Business Data Warehouse” [Devlin, 1988] that proceeds the actual “Data Warehouse” term. In 1991, Inmmon published the first edition of his book [Inmmon, 2002] where he defines the term “Data Warehouse”, and also consolidated the terms and techniques that have been the foundation for the Data Warehouse since then. In 1996, Kimball defined the Star and Multidimensional modelling techniques [Kimball, 1996] which enriched the Data Warehouse definitions.

Although relational databases were first believed that would provide direct access to data. It was very soon realised that they could not support both kind of workloads: Transactional (OLTP) and Decision Support (OLAP).

Transactional or OLTP systems typically consist of many users accessing small amounts of data, for example on an ATM bank system, there are hundreds of users accessing their account balances at the same time. Another difference is that such ATM system does not require keeping long periods of historical data; they only need the current balance.

In contrast Decision Support Systems (DSS) or OLAP can be accessed for few users but reading large amounts of data in order to make decisions. Using the same bank ATM system, the difference is that the people from the bank want to know the average balance for the last six months for the accounts with certain geographical region, in order to make strategic decisions, like opening a new branch office or encourage people to increase their investments by offering better interests. In order to solve this kind of query the system needs to keep historical data of the balances plus it would possible read millions of records of all clients within certain region and later on compute the average, this is very demanding access pattern which running on top of a Relational DBMS can lock large amounts of data and avoiding the Online users to gain access to their data. For such reasons the Data Warehouse has been separated in another copy of the database, typically on top of a RDBMS, but as it will be demonstrated in this thesis the traditional record orientation of the DBMS is not the best approach to manage large amounts of historical data.

Today, Data Warehousing usually involves the use of Multidimensional modelling of data, which is a data representation called *Dimensions* that arranges data, considering the different business views of those data [Kimball, 1996].

Data can be organized in different groupings or consolidation levels in each dimension called hierarchies [Agrawal, 1997b], for example, year, month, day is a time hierarchy and product name, type and category is a hierarchy for the product dimension. Inside each level, there are the actual member values or specific instances of the levels. For example, in the time dimension at the month level, the instance values correspond to the months of the year (January, February, March, and so forth).

Another important element of multidimensional modelling is called the *Fact Table* [Kimball, 1996]. Facts are the things that each organization wants to analyze, for example sales value. Facts are also known as measures [Agrawal, 1997b], also they are called numerical or summary attributes in the statistical database literature [Agrawal, 1997b].

In multidimensional there are a series of operations in order to manipulate data they are called: *pivoting* (rotate the cube to show a particular face), *slicing-dicing* (select some subset of the cube), roll-up (aggregate data to an upper level of a hierarchy), the converse is called drill-down (display detailed information for each aggregated point), in [Agrawal, 1997b] a set of algebraic operations has been defined in order to implement the previous multidimensional operations on top of relational systems.

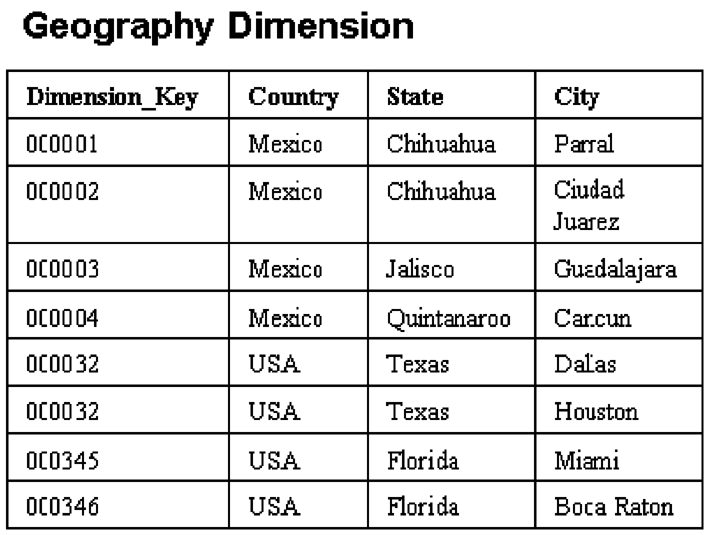
2.3.1 Data Warehouse Logical Design

Logical modelling of data is relevant in order to provide meaningful results to the users. There are two basic logical models in Data Warehousing: the Star and Snowflake [Microstrategy, 1999], which will be described briefly in the following sections.

The support of these two logical data models is an integral part of the proposed reference architectural configuration (ADW) (Chapter 8), in order to keep compatibility at a logical level with the current Data Warehousing installed base, owing to the vast knowledge of these modelling techniques, promoted mainly by Kimball [1996] and the large investment in modelling knowledge. ADW will support these modelling techniques by presenting tables at the logical level to the developers. It is necessary to differentiate the logical Data Warehouse design (star or snowflake) at the logical level, from the underlying data model on which the DBMS is based (Relational, Binary-Relational, Associative/Triple Store or Transrelational, are the ones considered in this study).

2.3.1.1 Star

This model is a non-normalized model; it means that each *Dimension* is represented by one table (Figure 1) that does not follow normalisation rules [Codd, 1990]. The star model has a *Fact table* in the middle of the model and has one table for each *Dimensio*n, see Figure 2.



Geography Dimension

Dimension_Key	Country	State	City
0C0001	Mexico	Chihuahua	Parral
0C0002	Mexico	Chihuahua	Ciudad Juarez
0C0003	Mexico	Jalisco	Guadalajara
0C0004	Mexico	Quintanaroo	Carcun
0C0032	USA	Texas	Dallas
0C0032	USA	Texas	Houston
0C0345	USA	Florida	Miami
0C0346	USA	Florida	Boca Raton

Figure 1 A Non-normalised Dimension

Following to the acceptance of the Star Model, specialised RDBMSs such as Redbrick [Redbrick, 2007] have been developed with the intention of optimising its performance under the Star model.

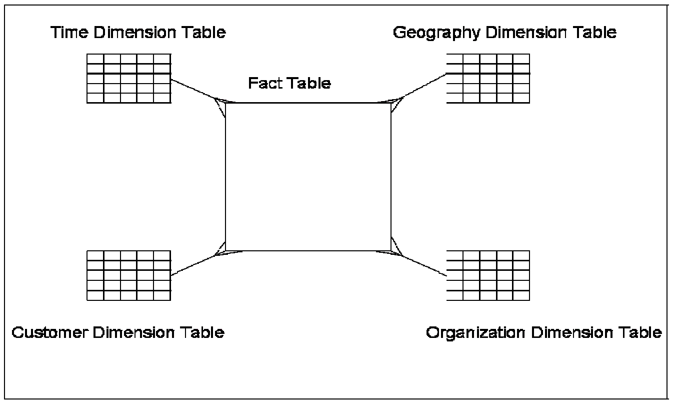
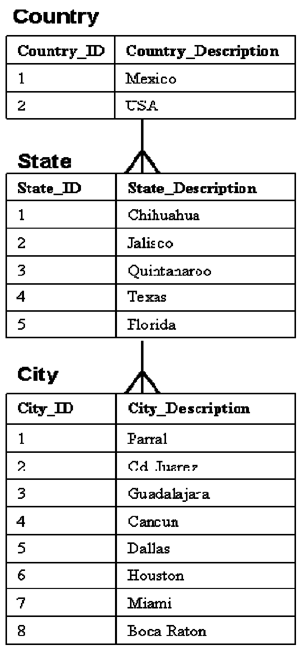


Figure 2: A Star Schema

2.3.1.2 Snowflake

The Snowflake model differs from the Star model in the sense that the Snowflake is a normalised model. This means that each *Dimension* follows Codd's normalisation rules. The *Dimension* is composed of a number of tables, where each table at one level has a many-to-one relationship with other tables at the next level, see Figure 3.



Country

Country_ID	Country_Description
1	Mexico
2	USA

State

State_ID	State_Description
1	Chihuahua
2	Jalisco
3	Quintanaroo
4	Texas
5	Florida

City

City_ID	City_Description
1	Parral
2	Cd Juarez
3	Guadalajara
4	Cancun
5	Dallas
6	Houston
7	Miami
8	Boca Raton

Figure 3 A Normalised Dimension

The Snowflake model has a central *Fact table* and uses many normalized tables for each dimension; one table for each dimension level as shown in Figure 4.

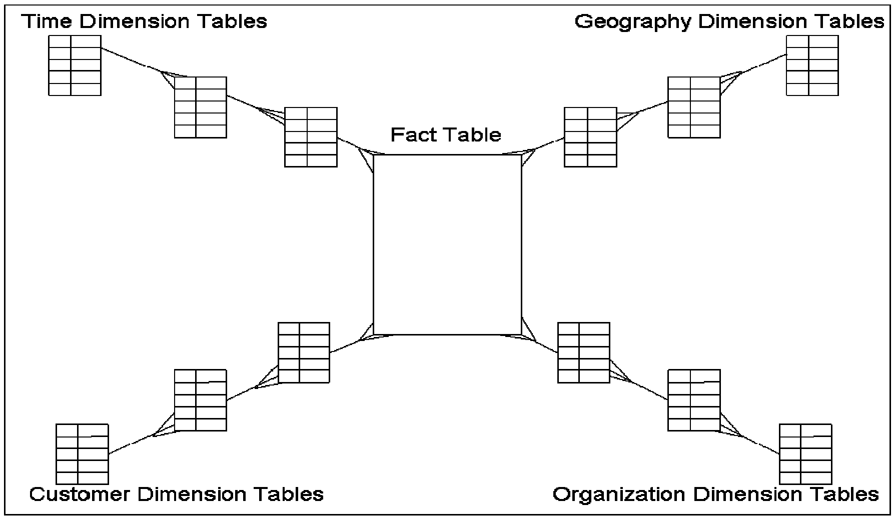

Time Dimension Tables
Geography Dimension Tables
Fact Table
Customer Dimension Tables
Organization Dimension Tables

Figure 4: A Snowflake Schema.

2.3.2 On Line Analytical Processing (OLAP) Approaches

The most common way to analyse data within a Data Warehouse is the one called OLAP and for this reason, existing OLAP approaches have been analysed. The term OLAP was coined by Codd to characterise the requirements for summarizing, consolidating, viewing, applying formulae to, synthesizing data according to multiple dimensions [Agrawal, 1997b]. Unfortunately, sometimes OLAP is misunderstood and the Data Warehouse appears as what OLAP can do, while the Data Warehouse is the data repository and OLAP is a technique to utilise such data.

2.3.2.1 OLAP Storage Structures

Multidimensional models can be stored in relational databases (RDBMS) or in Multidimensional databases (MDBS). Multidimensional modelling is the foundation for analysing data using OLAP techniques. OLAP techniques used against relational databases are called Relational On-Line Analytical Processing (ROLAP) [Pendse, 2001a]. If the analysis is performed against a Multidimensional Database (which typically this multidimensional databases store its data on specialised arrays [Dinter, 1998]) then it is called Multidimensional On-Line Analytical Processing (MOLAP) [Pendse, 2001a].

Other variants of these basic OLAP techniques are also available; for instance, HOLAP stands for Hybrid OLAP, which stores some data in Multidimensional structures

(sometimes called cubes). However, if the analysis requires data not available in the cube, HOLAP is able to query the Relational Data Warehouse using SQL.
A further example is DOLAP, which stands for Desktop OLAP. DOLAP stores the multidimensional structure on the user's desktop computer instead of the server therefore allowing the user to do some analysis on his/her own computer. Besides, the user even can be working disconnected from the main Data Warehouse. However, this approach can be dangerous because multiple versions of the "true" may be circulating within an organization if each user has his/her own copy of the data. Furthermore, the initial purpose of a Data Warehouse, which is "a unique version of the true", is contradicted.
The research of this thesis has been focused towards ROLAP systems, as those are the largest existing enterprise Data Warehouses and are the ones that struggle most to manage storage efficiently and have query performance problems. However a Data Warehouse can certainly be used to feed any of the other OLAP approaches (MOLAP, HOLAP and DOLAP), as usually these approaches use a subset of the data contained in the Data Warehouse.

2.4 Existing Approaches to Increase Query Performance in Data Warehousing Environments

Query performance is the main problem identified by users and by IT people, as this is the final result of the Data Warehouse Architecture. This is one of the most common complaints made by end users against Data Warehouse environments [Pendse, 2003a]. This aspect is one of the top research priorities to improve the query performance (objective 6) by proposing a novel Data Warehouse architectural configuration that can manage sparsity, increase data density and improve query performance.

Different techniques have been proposed with the intension of improving query performance. Some of these are analysed in the following sections.

2.4.1 Summarized Tables

A summary table is the most basic way to pre-compute values [Microstrategy, 1999]. In the context of this thesis, a summary table is such a table which keeps data based on aggregation of values. For example if data sales figures are kept at a daily level, examples of summary tables are tables which values of sales figures values are kept at state or country levels. Generally, the Data Warehouse administrator takes care of the summary tables, as manual intervention is required. The application should be sufficiently intelligent to redirect the query to the summary table. Some OLAP tools can use summary tables when those exist. For example, Microstrategy [Microstrategy, 1999] can do that by defining the summary tables in the metadata database, and then when its query engine finds a summary table that is useful in order to solve a query, it re-rewrites the query to use the summary table.

The pre-computing of summary tables is basically what the cube tools do (refer to section 2.4.4). They pre-compute all possible aggregations to all different levels of the *Dimensions*, and compute the Cartesian product of all dimensions and all different aggregation levels see (Figure 5).

This approach has the following drawbacks when computing the Cartesian product:

- High processing time is required.
- Many of the pre-computed tables/values will not be used
- It adds to the Database explosion phenomena [Pendse, 2003b]
- It leads to a low Data Density because it computes the Cartesian product and produces many Null values.

The main benefit of this approach is that the online query response time for the end-user can be fast because everything is already pre-computed.

Summary tables are defined and maintained by the Data Warehouse administrator.

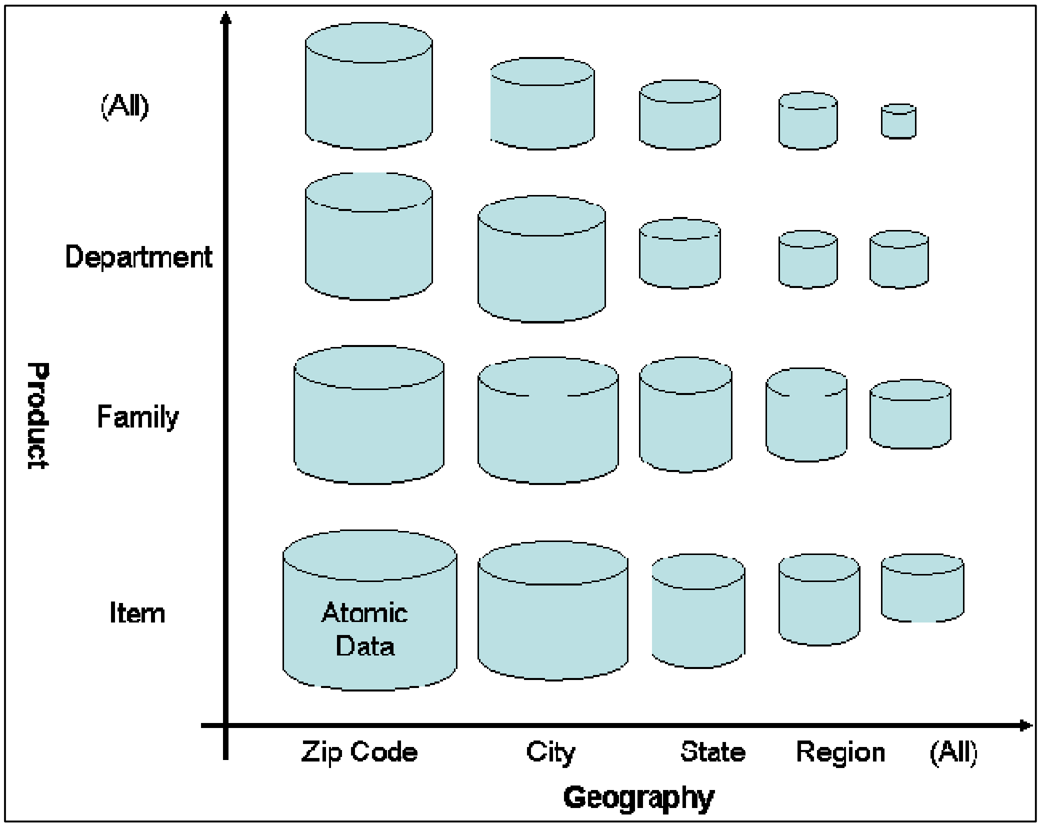


Figure 5 Cartesian Product Computation. From Microstrategy[1999]

2.4.2 Materialized Views

This is another approach to pre-computing values. It is basically the same as summary tables, but in this case the RDBMS takes care of maintaining and updating the materialized views [Akadia, 2007]. This approach has been followed by Oracle in its RDBMS [Akadia, 2007]. In early versions, they were called "Snapshots". Oracle RDBMS has incorporated the capability on the query optimizer to re-write queries in order to use materialized views when available [Akadia, 2007].

Summary tables and materialized views can help with query performance, but they have drawbacks:

- They require extra processing time
- They use extra disk space
- They need to be maintained (create indexes, compute statistics)
- Integrity constraints need to be managed

Even when materialized views could exist, applications need to be aware of them and in the majority of the time; programmers need to incorporate an extra code. In the case of Oracle, this code has been incorporated into the RDBMS [Akadia, 2007].

The consistency between the detailed table and summary tables can be compromised as a query can read any of the tables before the synchronization has been performed. It is worse in the case of summary tables than in the case of materialized views, as the RDBMS takes care of this functionality. When increasing the number of materialized views within a Data Warehouse, the window time available for maintaining materialized views is shrinking [Mistry, 2001].

The approach followed in this thesis does not suffer from these weaknesses as the proposition is to change the underlying model of the DBMS (objectives 3 and 4), which is the root cause of the problem. Hence, to increase the query speed (objective 6), it is not necessary to pre-compute results. However, as the proposed approach works on the underlying model of the RDBMS, the logical Data Warehouse model can still use summary tables or materialised views (objective 5 regarding keeping compatibility with the installed base) if required.

2.4.3 Approximate Queries

This is another approach to increasing query performance. In this case, instead of using all the tuples involved in the query and providing exact answers, only sample data are used, and then, using statistical methods the DBMS returns an approximate value plus the probability degree of the answer [Acharya, 1999]. People who follow this approach [Acharya, 1999; Shanmugasundaram, 1999] argue that, in order to make a decision, it is not necessary to process all the tuples, and, with a good enough result, the right decisions can be made.

In contrast, the approach followed in this thesis provides accurate query answers, as the DBMS uses all tuples but with better query times than the traditional n-ary-Relational based DBMS. Again, approximated queries work on the n-ary-Relational model.

2.4.4 Cubes

Hypercubes are the most commonly used structure in OLAP environments. Almost all OLAP tools are based on hypercubes -Cognos, Business Objects, etc. [Pendse, 2001a].

People refer to hyper-cubes simply as cubes. A hyper-cube is a single-cube logical structure. A hypercube is a storage structure that maintains data (generally, summarised results) and provides faster end-user query answers. Its main drawbacks are:

- Summarised results need to be pre-computed and it increases the batch processing time.
- If the query involves data which are not considered in the hypercube, then the cube is unable to answer the query, in contrast to ROLAP tools which access all detailed data. Hypercube vendors have been developing different approaches to minimise the limitation of answering queries, as Multicubes, Hybrid OLAP (HOLAP) tools and Iceberg cubes.

ROLAP tools that use only one Fact Table can also be considered Hyper-cubes because "cube data" are stored in a single Fact Table [Pendse, 2001a].

Multicubes: in this approach the application designer segments the database into a set of multidimensional structures, each of which is a subset of the overall number of dimensions in the database; this approach is used by products like Express, Pilot, Holos, TM1 and Microsoft OLAP Services [Pendse, 2001a]. Exponents of multicube systems emphasize their great versatility and potentially greater efficiency (particularly with sparse data) [Pendse, 2001a].

ROLAP products can also be logically multicubes if they can handle multiple base fact tables; for instance Microstrategy, Informix and Computer Associates' Information Advantage [Pendse, 2001a] are ROLAP products which can logically use multicubes by managing multiple fact-tables. In [Gray, 1996] it was proposed to extend the SQL language with a Data Cube operator to compute the cubes over the relational DBMS. Later on in [Agrawal, 1997b] a data model and a few algebraic operations are defined, these provided a semantic foundation to multidimensional databases but these proposed operations are easy to translate into relational algebra and therefore used by Relational DBMS.

During the research in thesis, a classification of cubes has been made, and it is presented in Figure 6 .

These cube approaches have been studied in order to increase query response times, which is part of the objective 6 in the thesis, but what has been found with the approach followed in this thesis is that using an architecture which uses an alternative data model on the DBMS (objective 4) provides faster queries times (objective 6) into the same Data Warehouse structure.

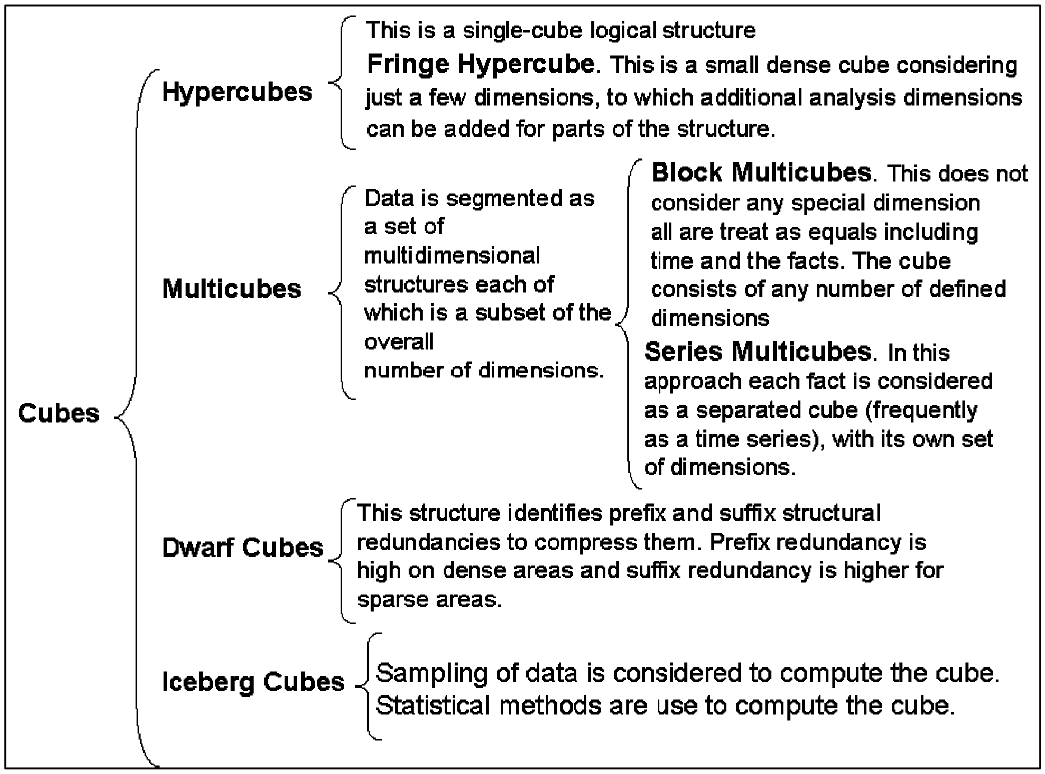

Cubes
Hypercubes
This is a single-cube logical structure
Fringe Hypercube. This is a small dense cube considering just a few dimensions, to which additional analysis dimensions can be added for parts of the structure.
Multicubes
Data is segmented as a set of multidimensional structures each of which is a subset of the overall number of dimensions.
Block Multicubes. This does not consider any special dimension all are treat as equals including time and the facts. The cube consists of any number of defined dimensions
Series Multicubes. In this approach each fact is considered as a separated cube (frequently as a time series), with its own set of dimensions.
Dwarf Cubes
This structure identifies prefix and suffix structural redundancies to compress them. Prefix redundancy is high on dense areas and suffix redundancy is higher for sparse areas.
Iceberg Cubes
Sampling of data is considered to compute the cube.
Statistical methods are use to compute the cube.

Figure 6 Cubes classification

Cube approaches store data on other data structures, which means duplicate data, and perhaps create "different versions of the true". With the approach followed in this thesis, a single "version of the True" is kept as it stores and processes every query into the Data Warehouse without duplicating data (objective 6).

2.5 Current Approaches to Increase Data Density

Objective 1 relates to the increase of data density. In this section an investigation of different approaches to increase data density has been made. The need to increase Data Density (number of records stored per disk unit) has existed since the beginning of the computer sciences. Traditionally it has been tackled with compression methods. For this reason, compression methods and their suitability in Data Warehouse environments have been investigated in this thesis.

2.5.1 Compression Methods

The result of minimising data sparsity and increasing data density is a smaller Data Warehouse. This Data Warehouse reduction can appear as a compression method, and, in order to identify similarities and differences between the approach followed in this thesis and compression methods, an investigation has been done on compression methods. The use of compression methods in databases was also considered.

According to Lelewer [1987], the aim of compression is to reduce redundancy in stored or communicated data, thus increasing effective data density. Data compression is of interest in business data processing. The types of local redundancy present in business data include runs of zeros in numeric fields, sequences of blanks in alphanumeric fields, and fields that are present in some records and null in others.

Run-length can be used to compress sequences of zeroes or blanks. Null suppression can be accomplished using presence bits.

Another class of compression methods exploits cases in which only a limited set of attribute values exists. Dictionary substitution entails replacing alphanumeric representations of information, such as bank account type, insurance policy type, sex, and month, by the few bits necessary to represent the limited number of possible attribute values [Lelewer, 1987].

In Table 1, a classification of the compression methods has been made according to their processing characteristics.

Table 1 A Classification of Compression Methods

Classification	Characteristics	Example
Static Code Words	Code Words are assigned according to the probability of the words appearance in the messages. Words that appear frequently are assigned to smaller code words. Requires two passes over the data set.	1. Static Huffman Coding. 2. Shannon-Fano Coding. 3. Universal codes
Dynamic Code Words Or Adaptive Methods	The mapping of code words changes over time. For this reason these methods are also referred to as adaptive methods. Requires one pass over the data set.	1. Dynamic Huffman Coding (FGK algorithm). 2. Lempel-Ziv 3. BSTW

2.5.2 Using Compression in Databases

The interest in using compression techniques in databases has been always present, because storage space savings have always been an issue to take into consideration. For example, in Cormack [1985], a general purpose data compression routine was presented. It was implemented in IBM's IMS Database Management System. This routine used a modified Huffman Coding, and it was useful and beneficial in terms of storage space.

Later in Westmann [2000], the intention was to demonstrate how compression can be integrated into relational DBMS and how the storage manager, the execution engine and the query optimizer can be extended to deal with compression. The TPC-D benchmark was used; the results presented gain up to 50% on I/O intensive queries and moderate gains on CPU intensive queries. The recommendation was to extend the RDBMs with lightweight compression techniques.

It has been recognized that compression can be the cause of significant CPU overhead to compress data the first time and to uncompress data every time it is used, as was done in IBM/IMS [Cormack, 1985] where it compresses individual records when input. The expansion is performed each time a record is retrieved. Therefore, as many relational applications execute CPU intensive operations (e.g. joins and aggregations), compression has not yet achieved wide acceptance in relational DBMS [Westmann, 2000].

Chen *et al.* [Chen, 2001] remarked that work has been done on the compression of numerical attributes, but not on the string-valued attributes in relational schemas. Here a lightweight, attribute-level based compression technique is presented. This technique considers the effects of the Query Optimizer that can produce sub-optimal plans for queries involving compressed string attributes. This approach contradicts Westmann's approach: "Compression could be integrated into a database system without any adjustments to the query execution engine and query optimizer of a database system".

In Chen [2001], it is also recognized that compression has traditionally not been used in commercial Database systems because many compression techniques are effective only on large chunks of data and are thus incompatible with Random Access to small parts of the data as they occur in OLTPs. These studies have been done mainly in OLTP databases. Nevertheless considering that data is chunked within Data Warehouse environments, Oracle RDBMS has introduced a compression feature in its DBMS version 9 [Poss, 2003] with the intention of compressing data when using Oracle in Data Warehouse environments. They use a lossless dictionary-based approach in which dictionary entries are created by the system. Their compression method is an adaptive method. It compresses duplicate data at Oracle's block level, which can produce suboptimal compression if the complete database is considered, but they report good enough compression ratios, 29% space savings in the TPC-H schema and 67% on a star schema [Poss, 2003].

Other DBMS, such as Teradata and DB/2, have incorporated compression methods [Morri, 2002; Ahuja2006].

The works presented in Goldstein [2000] and Westmann [2000] show that compression can improve performance in read-intensive environments such as Data Warehouses. They use the TPC-H benchmark, in which the schema contains 61 attributes, out of which 26 are string valued, constituting 60% of the total size of the Database. It is surprising that there has been little work in the database literature on compressing string attributes. Classic compression methods such as Huffman, Arithmetic coding, Lempel-Ziv and order preserving methods [Lelewer, 1987], all have considerable CPU overhead that offsets the performance gains of reduced I/O. Therefore, the use of compression methods in DBMS seems infeasible. Hence, existing work in the database literature employs simple, lightweight techniques such as Null suppression and dictionary encoding [Chen, 2001].

An important issue in compressed database systems is when to decompress the data during query execution. There are two basic approaches:

- Eager decompression: Data is decompressed when it is brought to the main memory. Eager decompression has the advantage of limiting the code changes caused by compression to the storage manager. However, eager decompression generates sub optimal plans because it does not take advantage of the fact that many operations such as projection and equi-joins can be executed directly on compressed data [Chen, 2001].

- Lazy decompression. Here data stay compressed during query execution as long as possible and are decompressed when necessary. However, this decompression can increase the size of intermediate results, and thus increase the I/O of later operations, such as sort and hashing, in the query plan [Chen, 2001].

In this thesis, a different approach has been made: change the underlying DBMS model instead of using compression methods on top of the RDBMS. This approach achieves the database reduction by avoiding repetitions at column level, and it does not pay the CPU processing overhead required by the compression methods (see chapters 6 and 7).

Also, with the approach followed in this thesis, all attribute types get benefits as the underlying data model it does not differentiate between data types; it avoids duplicates at column level; therefore the asseverations made by Goldstein [2000], Westmann [2000] and O'Connel [2003], have been tackled. In O'Connel [2003], another point is made: "There has been much work on compressing database indexes, but less on compressing the data itself". The algorithm presented in that research work enables the processing of queries without decompressing the data needed to perform join operations in a database built on a triple store. Records are compressed when initially inserted into the triple store, but from then on, processing can be carried out efficiently without needing to decompress the records again.

As O'Connell used a Triple Store, then this data model has been one of the alternative data models studied during the research for this thesis (section 3.3)

2.6 A Relational Data Warehousing Architecture

Relational model and relational technology are the paradigm most frequently used in the DBMS industry today. Since the inception of the Data Warehouse concept in Devlin [1988], the use of a relational DBMS (DB/2) was considered. The initial conception of a Data Warehousing Architecture was created in order to satisfy internal IBM Decision Support Analysis in Europe (see Figure 7).

Later on vendors like Platinum suggested an architecture considering RDBMS [Gill1996] as the technology to build Data Warehouses as can be seen in Figure 8.

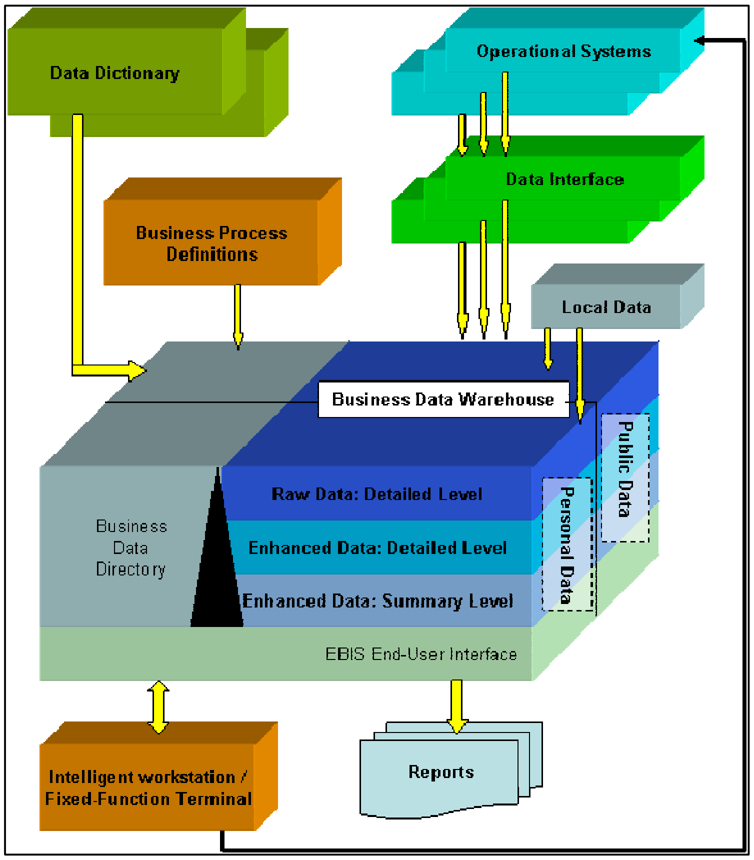


Figure 7 Initial Architecture of a Business Data Warehouse. From Devlin [1988]

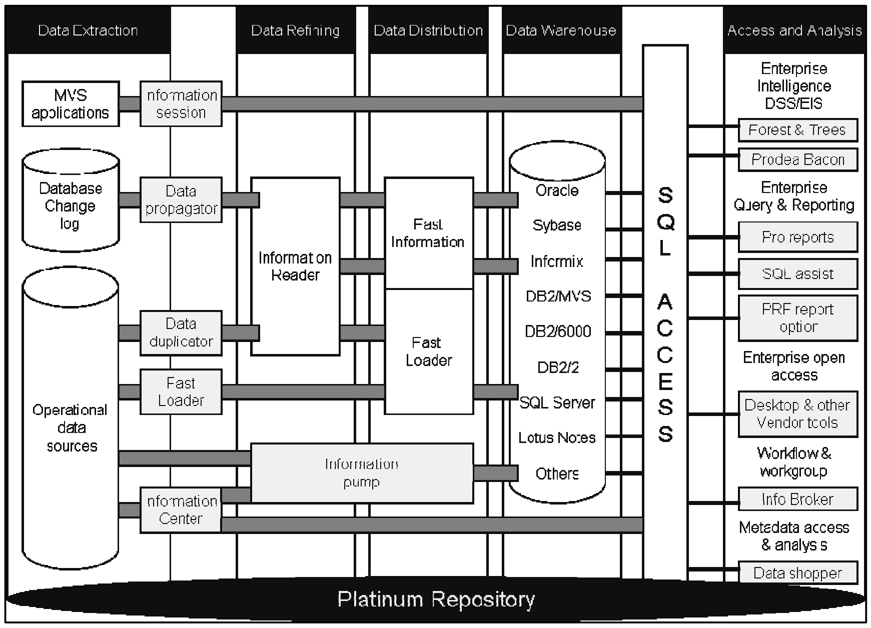


Figure 8 Platinum's Data Warehouse Architecture. From Gill [1996]

Therefore, a generic Relational Data Warehouse Architecture will include different components:

- Transactional Systems or Operational Systems
- An Extraction-Transformation and Load Stage
- A Relational Data Warehouse Repository
- Application Tools (Report Writers, ROLAP tools, CRM tools, Data Mining Tools, MOLAP tools, which can include Multidimensional Databases as Essbase [Elkins, 1998])

In this thesis, this architecture is referred to as "Traditional Relational Data Warehousing Architecture" and is represented in Figure 9.

Research objectives 1 to 6 are focused on finding a data model better suited to substitute the relational Data Warehouse. Research on non-Relational models has been carried out and is presented mainly in Chapter 3.

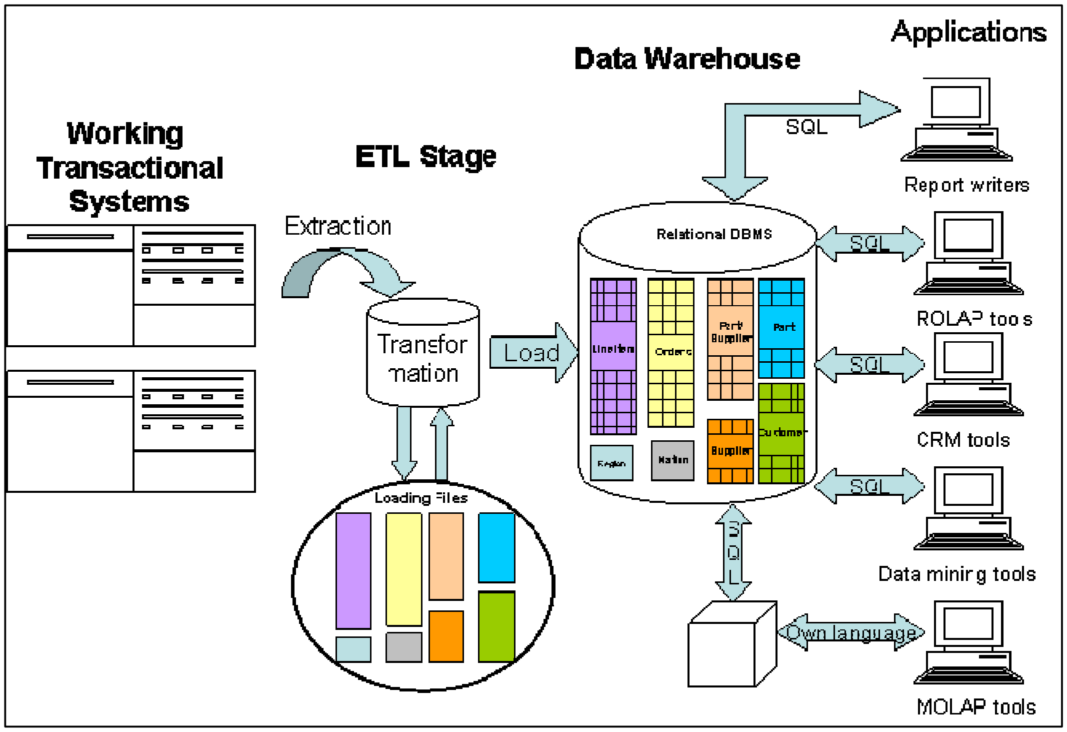


Figure 9 Traditional Relational Data Warehouse Architecture

2.7 Conclusions

While studying how to address the issue of existing data sparsity in Data Warehouses, the weakness of the relational model in managing missing data is a key factor that affects sparsity. Data Sparsity is closely allied to the issue of Data Density, because sparse data has low data density.

In Data Warehouse environments, many repetitive values are stored at a column level when using an n-ary-relational approach; therefore data density can be increased by storing just one instance of each value and using it as many times as required. The n-ary-relational model is not capable of doing that, because the linkage structure of the data within a row is made by its physical contiguity. The alternative data models considered in this thesis separate the data values from its linkage structure.

From the analysis of the existing solutions to the problem (summary tables, materialised views, approximate queries, cubes and compression methods), none of which offered any major improvements, all the proposed solutions are based on extensions of the prevalent n-ary-relational model and they do not solve the root cause of the problem. From this it has been decided to investigate the potential of non n-ary-relational models, which have been called in this thesis ‘alternative data models’ this being one of the contributions of the current research. Another contribution of this thesis is the explicit cubes classification made in this chapter.

CHAPTER 3
Alternative Data Models

While researching the data sparsity and data density issues, a group of non relational data models were investigated in order to understand their behaviour in Data Warehouse environments. The selected models are:

1. The Associative/Triple Store model [Williams, 2003; Sharman, 1988]
2. The Binary-Relational model [Titman, 1974; Copeland, 1985; MonetDB, 2004a].
3. The Transrelational model [Date, 2004]

The characteristics of each model are presented in this chapter, and their benchmarking results are presented in Chapter 6 and analysed in Chapter 7. In this thesis, these models are referred to as Alternative Data Models.

3.1 Relational, the Base Model

The relational model - or more strictly speaking the n-ary-Relational model - is the base model against which the alternative data models have been compared and benchmarked. This is because Relational model and Relational technology are the most widely adopted technology in the world.

3.1.1 Behaviour and Characteristics

According to Codd's Relational Model [Codd, 1970], all operations are made using the mathematical concepts of sets. For this reason every operation is made – conceptually - in sets. Current implementations [IBM, 2008a; Oracle, 2008; Microsoft, 2008; Sybase, 2008] appear to the user as if they were processing sets, but in reality it is not how the underlying implementation works, as it continues processing internally row by row [Date, 2005], following a horizontal approach to store and manage data.

3.1.2 Storage Structures

The storage structure is the *Relation* or in Date's nomenclature "*Base Relations*" which conceptually hold all information. These conceptual relations are known in relational products as *tables*, but strictly speaking a *table* is not a Relation [Date, 2005; Codd, 1990]. An example of a *Relational Table* from [Date, 2004] is presented in Figure 10. As can be observed, duplication exists at column level, which reduces the data density.

P#	PNAME	COLOUR	CITY
P1	Nut	Red	London
P2	Bolt	Green	Paris
P3	Screw	Blue	Oslo
P4	Screw	Red	London
P5	Cam	Blue	Paris
P6	Cog	Red	London

Figure 10 A Relational Table. From Date [2004]

3.1.3 Model Instantiation

Oracle version 9i was the n-ary-Relational model instantiation used in this research.

3.2 Abandoning the Record Storage and Processing Structure

In this thesis, a group of alternative data models (Associative/Triple Store, Binary-Relational and Transrelational) has been chosen, because they abandon the record storage and processing structure. They store data vertically and eliminate duplicates at column level by storing each value just once in a storage structure; then they use this value wherever required by relating it to many records via linkage structures. By doing this, it can be stated that: *"Following a vertical approach and storing values just once with in the storage structures the data density will be increased and the final Data Warehouse size will be reduced"*.

3.3 The Associative/Triple Store model

The triple store model has its foundation in Sharman's seminal paper "The Universal Triple Machine: a Reduced Instruction Set Repository Manager" [Sharman, 1988], in which the concepts of the model are introduced. It is argued that triplets can be used to construct many systems covering a wide range of information, and bearing structures such as First Order Logic formulae, PROLOG programs, semantic nets, conceptual graphs, binary relationships and normalised Database records. In this research, one of the directives was to use and benchmark triplets to store Data Warehouses.

In Sharman [1988] it is also stated that “Database workers have tended to rely on familiar record structures (without exploring the limit conditions where these structures prove to be inadequate)”. This is precisely the situation that has been researched in this thesis, because record structures are not the best structure in which to store and process Data Warehouses [Gonzalez, 2004].

The Triple Store continues to be a subject of research [O’Connel, 2003]. The TriStarp project [TriStarp, 2000] has considered Triple Store. Also in Gray [2004] the consideration to use the Triple Store model to implement functional languages has been made. IBM has been investigating the use of triple store to manage Databases [TriStarp, 2000].

In Williams [2003] the argument that adding a 4th element to the linkage structure (the triple store) gives more flexibility and power to the data model, and, instead of calling it Triple Store, it is called *Links*. In Williams [2003] the Associative model of data is presented. It uses Quadruplets instead of Triplets, but essentially they are based on the same concepts and for that reason they have been grouped and referred to in this research as the “Associative/Triple Store Data Model”. These models have been used in the implementation of functional languages and inference systems as they were made for these kind of systems.

3.3.1 Behaviour and Characteristics

According to Williams [2003], databases that use the Associative Model store all different types of data in one single logical structure (the name store), while relational databases use unique structured tables to store each different type of data (Relations). In section 3.3.2, the storage structures are presented for both Triple Store and Associative models.

A possible weak point of these models is that names are not checked for uniqueness at the time they are added to the store, since this would imply some semantic associated with them. Once names are added to the store, they can be retrieved only via their identifiers. This means that each identifier must be “remembered” (in the triple store); otherwise its corresponding name will be lost forever [Sharman, 1988]. To preserve the integrity of the mapping, identifiers must be unique, and one should never re-use identifiers [Sharman, 1988].

The Triple Store does not rely on any particular interpretation of a triple. However, one common interpretation is that a triple corresponds to an elementary fact of the form:

<t1,P,t2>

Where P is a predicate symbol and t1 and t2 are terms [Sharman, 1988]; that is why Triple Store implementations are used by functional languages, as they are based on predicates. In Gray [2004], its use in functional languages is studied.

According to Sharman [1988], implementations of the triple store will always order the triples, partially or fully, for two important reasons:

- Ordering allows efficient implementation of the INSERT_TRIPLE, DELETE_TRIPLE and FIND_TRIPLE operations.
- Most programming languages cannot accommodate operations which return a set as a result, and require results to be returned one at a time; and having the triplets ordered facilitates the set operations that can be made by functional languages.

A number of logical data models may be implemented in terms of triples. For record-oriented data models, this usually requires that each field in an n-ary record be represented by a triple. Hence *n* triples are needed to represent an n-ary record. In Figure 11, the representation of a generic *Parts* entity is made when using the Associative/Triple Store model. For comparison purposes, a Relational representation is also made.

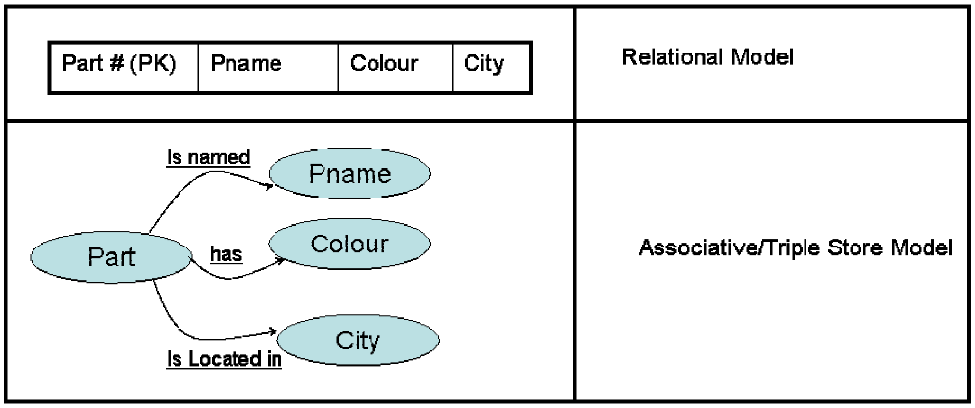


Figure 11 Logical Model of the Parts Entity in Relational and Associative/Triple Store

Some basic requirements for any implementation of the triple machine, as defined in Sharman [1988], are:

- Large data capacity
- Persistent memory
- High Performance

According to Sharman, [1988], some systems that used Triple Store have been in continuous production for many years without database re-organisation. This behaviour is attractive for a Data Warehouse environment, as these systems never would require a reorganisation operation, which is one of the factors which contributes to the Data Warehouse exponential growth; it is also one of the motivations for the research carried out in this thesis (section 1.4.1). An example of an n-ary Relational implementation which does not fragment data is Software AG's ADABAS-D DBMS, which avoids reorganisation of the data space as it is continuously moving records within the data space to recover fragmented space and to balance the workload over the disks [SoftwareAG, 1997a].

An approximately linear relationship between the number of triples and physical database size has been observed by Sharman [1988]; it has also been observed in the experiment carried out in this thesis (see section 6.4.3.1).

Names typically account for a small proportion of the total storage requirement, since they are not duplicated [Sharman, 1988]. This behaviour has been verified in the experiments of this research; furthermore, the same behaviour has occurred in the other models (see sections 5.5, Chapter 6 and Chapter 7) in which the names or descriptions count for a small part of the database size, but the linkage structures occupied the larger amount of space, as highlighted in section 5.5. Hence special attention should be given to the models. At first sight, they appear to eliminate duplicates and reduce space by this method, but with the linkage structures, those space gains can vanish (see section 5.5).

Codd in [Codd, 1970] states that "Future users of large data banks must be protected from having to know how the data is organized in the machine". This was a step forward in allowing programmers to use their time more productively [Williams, 2003]. But the work presented in Williams [2003] goes further with the Associative model by stating that "The aim of the relational model was to free programmers from having to know the physical structure of data; the aim of the Associative Model is to free them in addition from, having to know its logical structure".

In Williams [2003], a comparison of the building blocks of the Associative model against those of the relational model is made as follows:

- An entity type that is the source of one or more association types is equivalent to a table that has no foreign keys as primary keys.
- An entity type that is the source of no association type is equivalent to a domain.
- An association type is equivalent to a table that has one or more foreign keys as primary keys.
- An association type's verb is equivalent to a column heading in a relation.

Those modelling equivalences have been applied in order to obtain the TPC-H logical model under the Associative/Triple Store perspective. It is presented in Appendix C. Such a model has been built and loaded with data, and its corresponding benchmarking results are presented in Chapter 6 and analysed in Chapter 7. The associative model does not use records [Williams, 2003].

3.3.2 Storage Structures

In Sharman [1988], the Triple Store Model's structure is defined as consisting of two basic storage structures:

• *The Name Store*: It stores all the information related to the "real world" names. It is a structure with two columns: one for the names and the other for the unique identifiers which are generated automatically (Figure 12(a)). There is no differentiation in the information stored; the repository must be able to store these names as lexical objects.

• *The Triple Store:* It stores the *triplets* that are ordered 3-tuples, <i1, i2, i3>; where each element of the triple is an identifier which must have been generated by the name store before the triple is created (Figure 12(b)). The information required to rebuild the record is kept here [Sharmman, 1988].

1	Nut
2	Red
3	London
4	Bolt
5	Green
6	Paris
7	Screw
8	Blue
9	Oslo
10	Cam
11	Cog

1	2	3
4	5	6
7	8	9
7	2	3
10	8	6
11	2	3

Figure 12 Physical (a) Name Store and (b) Triple Store

According to Williams [2003], the Associative Model of data improves the Triple Store, by adding an extra element and using quadruplets instead of triplets. The names of the structures are changed to:

•*Items*, each of which has a unique identifier, a name and a type (Figure 13(a)).

•*Links*, each of which has a unique identifier together with the unique identifiers of three other things: the source, verb, and target (Figure 13(b)).

The *Items* structure keeps all the data. It stores unique values and relates these as many times as required via the *Links* to reconstruct the record.

1	Nut
2	Red
3	London
4	Bolt
5	Green
6	Paris
7	Screw
8	Blue
9	Oslo
10	Cam
11	Cog
12	Has Colour
13	Is Located in

100	1	12	2
101	100	13	3
102	4	12	5
103	102	13	6
104	7	12	8
105	104	13	9
106	7	12	2
107	106	13	3
108	10	12	8
109	108	13	6
110	11	12	2
111	110	13	3

Figure 13 Physical (a) Items and (b) Links

3.3.3 Model Instantiation

The model instantiation used for this research has been SentencesDB Version 3.5.

3.4 The Binary-Relational Model

According to Williams [2003], Chen's [1976] paper "The Entity-Relationship Model – Towards a Unified view of Data" [Chen, 1976] is widely credited as the origin of the Binary-Relational model, although there was a considerable amount of earlier work [Feldman, 1965; Levien, 1967; Ash, 1968; Feldman, 1969; Titman, 1974]. Even in Titman [1974], a prototype of a Binary-Relational system was created.

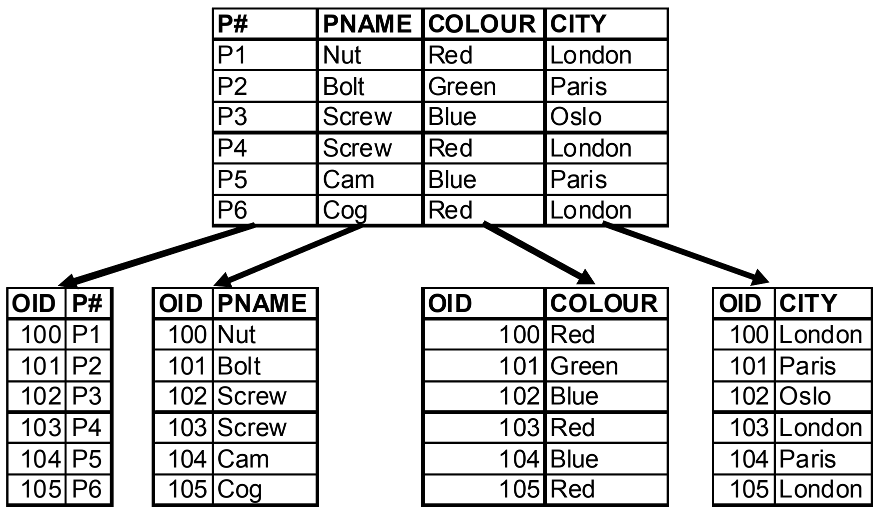
Recent work using the Binary-Relational Model includes that of Boncz [2002], Teklitz [2003], MonetDB [2004a], Gonzalez [2005b], Sybase [2005a], Stonebraker [2005], Zukowski [2005], and Stonebraker [2007].

The Binary-Relational model can be considered a particularization of the Relational model, as both are based on the mathematical concepts of relations. The Relational Model uses the generalization of use relations of *degree "n"* meaning that relations can have n-attributes. This is why it is referred to in this thesis as the "*n-ary-Relational Model*". By in contrast, in the Binary-Relational model all relations are Binary relations (relations of degree=2), meaning that all relations have two elements, one surrogate key plus an attribute. This is why it is referred to in this thesis as the "*Binary-Relational Model*".

The Binary-Relational Model is not new in the computer science body of knowledge, but its use in Data Warehouses certainly is. In this research, the measurement and benchmarking of the Binary-Relational model have been done to investigate its behaviour within the context of Data Warehousing.

3.4.1 Behaviour and Characteristics

The basic idea is that all the tables in the system are Binary tables with one surrogate key (it has been called OID) plus an attribute. To rebuild the record, it is necessary to execute *joins* between the binary tables, see Figure 14.



P#	PNAME	COLOUR	CITY
P1	Nut	Red	London
P2	Bolt	Green	Paris
P3	Screw	Blue	Oslo
P4	Screw	Red	London
P5	Cam	Blue	Paris
P6	Cog	Red	London

OID	P#
100	P1
101	P2
102	P3
103	P4
104	P5
105	P6

OID	PNAME
100	Nut
101	Bolt
102	Screw
103	Screw
104	Cam
105	Cog

OID	COLOUR
100	Red
101	Green
102	Blue
103	Red
104	Blue
105	Red

OID	CITY
100	London
101	Paris
102	Oslo
103	London
104	Paris
105	London

Figure 14 Binary Model's Table Decomposition

In Copeland [1985], the idea of a vertical fragmentation was introduced under the name of "A Decomposition Storage Model" splitting a table into a set of binary relations. This idea is implemented now in some RDBMS: MonetDB [MonetDB, 2004a], Sybase IQ [Sybase, 2005a] and C-store [Stonebraker, 2005] in which each has a decomposition storage model based on binary tables.

The research carried out during this thesis has been done using MonetDB, and the results are reported in chapters 6 and 7. Some other experience has also been reported, using Sybase IQ, in section 8.5.

In the next section, the logical structures of the binary-relational model are explained and the physical storage structures of MonetDB in particular are presented.

3.4.2 Storage Structures

Conceptually the logical storage structures of the Binary-Relational Model are binary tables which consist of one surrogate key plus one attribute, as can be observed in Figure 14. On the other hand, MonetDB's physical implementation is based on structures called BATs (Binary Association Tables) [Boncz, 2002; Kersten, 2005]. In Figure 15 a BAT structure is presented.

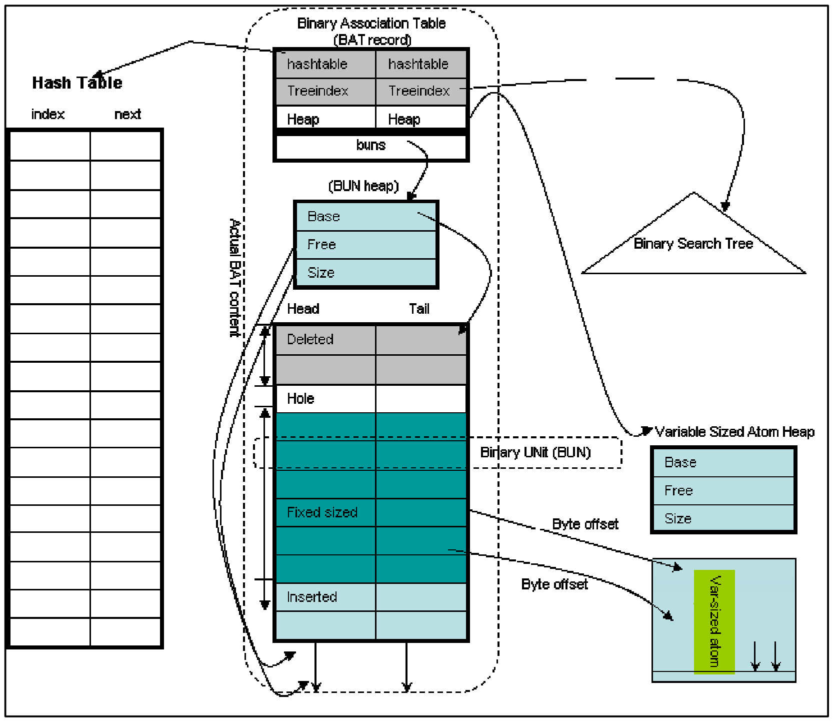


Figure 15 A BAT structure. From Kersten [2005]

The Binary-Relational representation as illustrated on Figure 14 is the way Codd [1990] has modelled the Binary-Relational, but in this way it is useless for the purposes of the current research as instead of decreasing the Data Warehouse size it will grow and nearly doubling the size of the Data Base (see analytical model's estimates in section 7.3.3.1); therefore improvements to the Binary-Relational Model have been introduced to eliminate duplicates. MonetDB uses BAT structures are used to optimise storage by eliminating duplicates but keeping track of which records are using the different values; it is made by the BUN Heap structures; if not, then the correspondence will be lost. BATs include OIDs which are object identifiers which help to keep track which value is used by which records. On Figure 16 the duplicates elimination process is illustrated.

According to Kersten [2005], "A BAT is a contiguous area of fixed-length records with automatically maintained search accelerators. Variable length elements are collected in separate storage areas, called heaps. New elements are appended and elements to be deleted are moved to the front until the transaction commits. The BATs are memory mapped from disk, and the memory management unit of the system is used to guarantee transaction isolation".

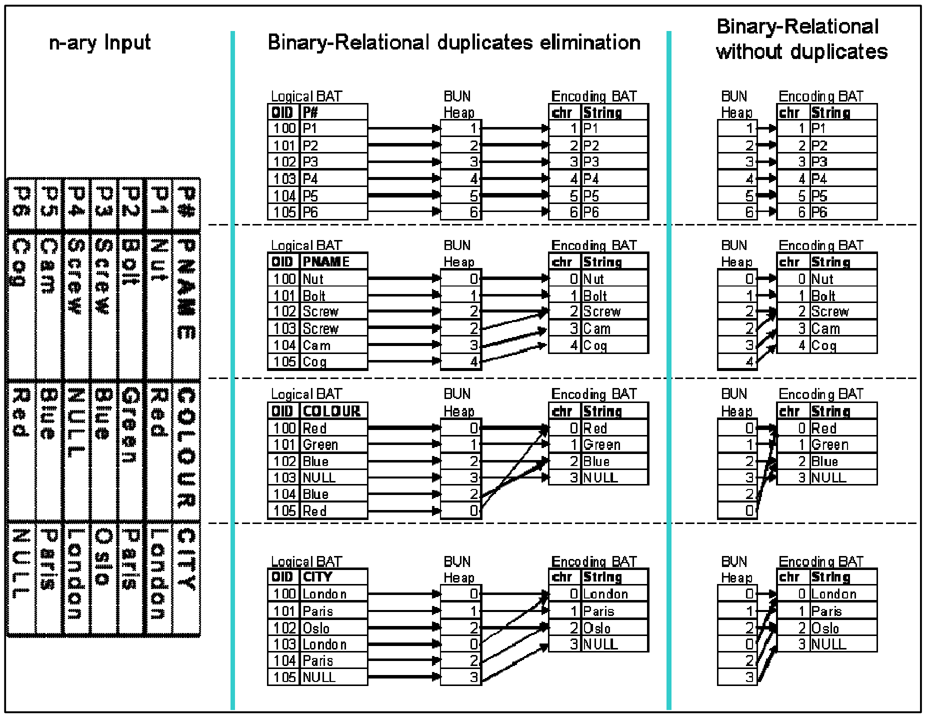


Figure 16 Binary-Relational Duplicates Elimination

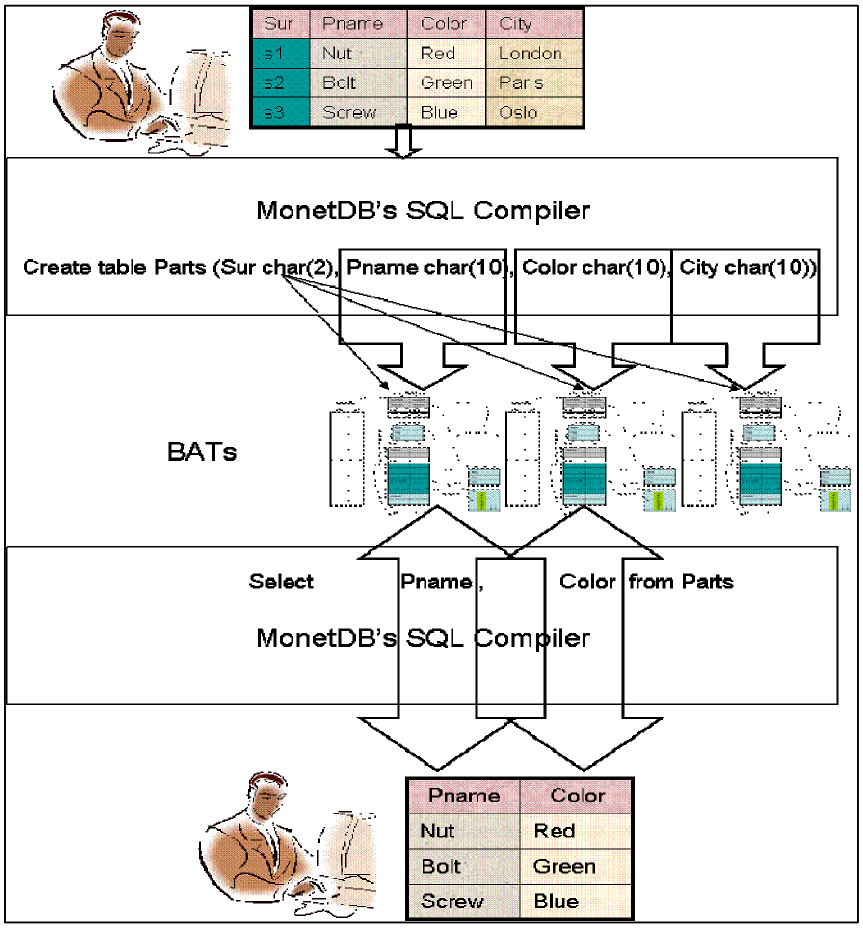
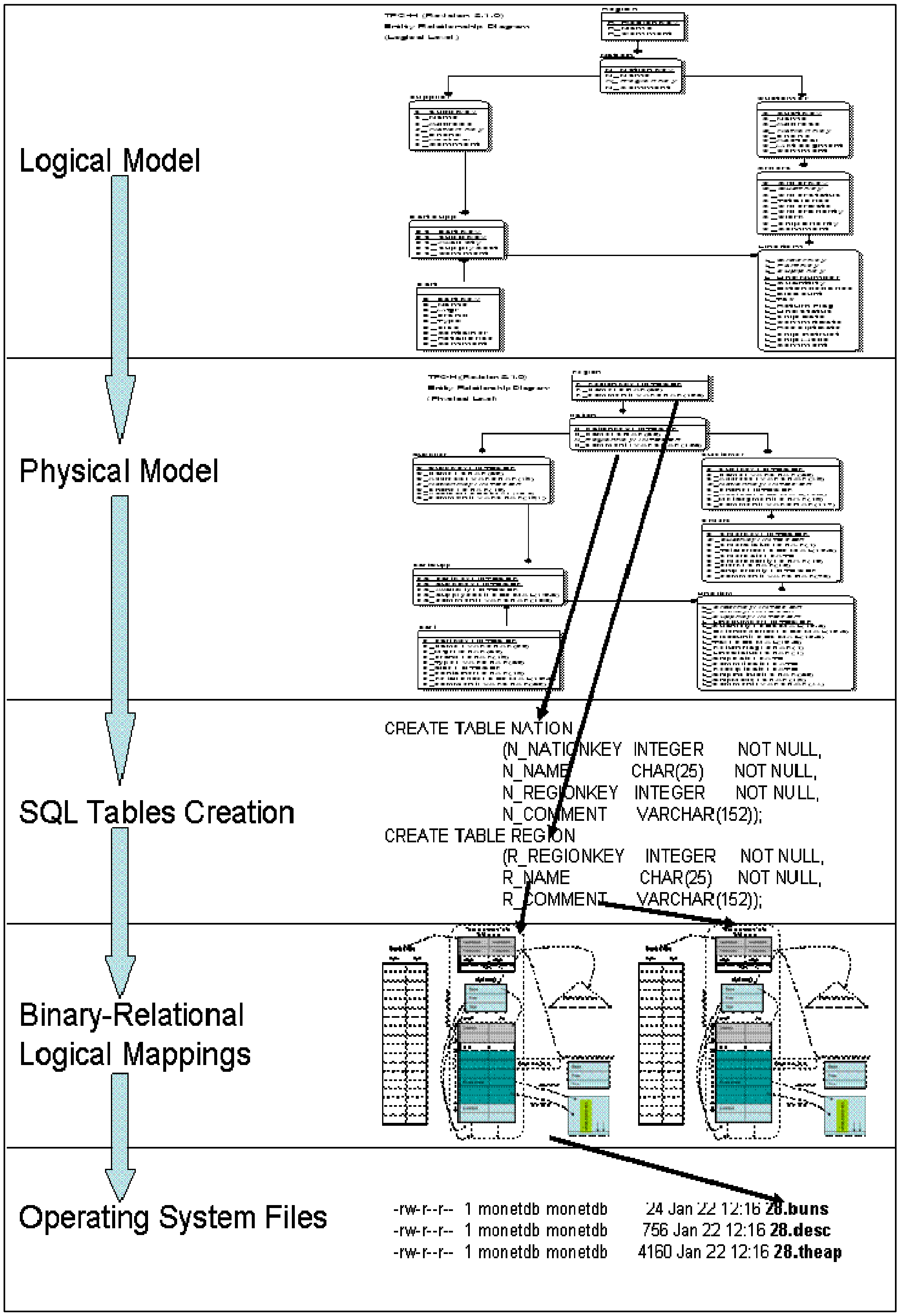


Figure 17 MonetDB's SQL Compiler mappings

It is important to appreciate that the Data Warehouse logical model is able to use the Star, Snowflake schemas or even Entity Relationship diagrams; it does not change. The Binary-Relational model is used for the RDBMS design, and finally the physical implementation is made using BATs which are stored in files at the operating system level (see Figure 18 to have better understanding of the different modelling levels).

It starts with the creation of a logical model which models the business needs; after that a physical model is made in which data types, column lengths and constraints are added. The next step is to execute the SQL statements in order to create the physical model into the DBMS. In this case, MonetDB's SQL compiler maps each table to a series of Binary-Relations (BATs). Those BATs are stored physically in operating system files. Each BAT is composed of three physical files with extensions *.buns*, *.desc, and .theap.* This process is illustrated in Figure 18.



```
CREATE TABLE NATION
                 (N_NATIONKEY  INTEGER      NOT NULL,
                  N_NAME       CHAR(25)     NOT NULL,
                  N_REGIONKEY  INTEGER      NOT NULL,
                  N_COMMENT    VARCHAR(152));
CREATE TABLE REGION
                 (R_REGIONKEY  INTEGER      NOT NULL,
                  R_NAME       CHAR(25)     NOT NULL,
                  R_COMMENT    VARCHAR(152));
```

```
-rw-r--r--  1 monetdb monetdb    24 Jan 22 12:16 28.buns
-rw-r--r--  1 monetdb monetdb   756 Jan 22 12:16 28.desc
-rw-r--r--  1 monetdb monetdb  4160 Jan 22 12:16 28.theap
```

Figure 18 Logical and Physical Modelling Levels

3.4.3 Model Instantiation

The model instantiation used was MonetDB Version 4.4.0 DBMS, developed at CWI, the National Research Institute for Mathematics and Computer Science of The Netherlands [CWI, 2008].

3.5 The Transrelational^{TM} Model

The Transrelational Model is based on an algorithm designed by Steve Tarin and patented in the United States of America [US Patent, 1999], but has been mainly promoted by C.J. Date in his book [Date, 2004] and with a series of seminars. In Chapter 5 of this thesis, the Transrelational model is studied in grater detail. At this point, only its main characteristics are mentioned in order to allow comparison of the main elements of each of the alternative data models.

3.5.1 Behaviour and Characteristics

The Transrelational model does not try to replace the Relational model; instead it works at a lower level, closer to the physical storage. The idea of Transrelational is to separate the values and the linkage information that tie those values together. In a direct image system *DIS* (record structure), both parts are together because the linkage information is represented by physical contiguity, while in Transrelational they are kept separate [Date, 2004].

The way Transrelational organizes data will "compress" information, compared with a traditional implementation of the relational model within the current DBMS. The implementation *per se* eliminates duplicates at the column level by the *Column Condensation Process* (Section 5.2 and Figure 31). This column condensation process will help to eliminate duplicates and increase Data Density, which is one of the major objectives of this thesis.

Once that information is loaded, it is simple to formulate queries and also many operations are no longer needed. For example, the internal sort in an ORDER BY clause is not needed [Date, 2004] as the data is already sorted. This implies less CPU processing time, less memory allocation, less execution time and fewer temporal disk areas.

The model itself is "true domain oriented" [Date, 2004], which is another major criticism of Codd on all RDBMS implementations. This is achieved because only different values are stored at column level. Hence all values of the domain are represented within each column, and it accepts data of one data type only. This characteristic was also found in the Binary-Relational Model.

Indexes are no longer needed, because each column stores different values only and each column can be treated as an independent object. When observing the advanced version of the *Field Values Table* on Figure 19, each column can be considered an index because an index keeps the different values sorted by specific criteria. This characteristic was also observed in the Binary-Relational Model. By avoiding the creation and management of indexes the following benefits are achieved:

- Disk space savings.
- Less CPU consumption.
- Fewer data structures to maintain.
- The query optimizer will be simplest or even non-existent.
- The work of the DBA will be less in designing and maintaining indexes.

According to Date [2004], the join times are linear because Transrelational always does a sort/merge join to solve the *join* relational operator but the sort and merge are done ahead of time, therefore the run-time join cost are additive (linear) not multiplicative as they are on current RDBMS where different join strategies can be follow (nested loops, Index lookup, Hash lookup, Merge and Hash) [Date, 2004]. Count operations can be computed directly from the ranges stored on the data itself; it is not necessary to read and count each row [Date, 2004]. Count operations are common in Data Warehouse environments. These characteristics are the main ones that made this alternative data model interesting for this thesis.

3.5.2 Storage Structures

The Transrelational model has two structures plus an algorithm to rebuild the records:

•**The Field Values Table**. Each column of the table contains the values from the corresponding field of the file, rearranged into *ascending sort order* (Figure 19) and in an advanced version of the Field Values Table without duplicate values in each column.

•**The Record Reconstruction Table**. The cells are the corresponding row numbers from the original table (Figure 20) but arranged in certain order determined by the Transrelational algorithms [Date, 2004]. This structure is used to reconstruct the record using again the Zigzag or ring algorithm as described in Date [2004] and implemented in section 5.3; with this algorithm the meaning and structure of this table is understood.

P#	PNAME	COLOUR	CITY
P1	Bolt	Blue	London
P2	Cam	Blue	London
P3	Cog	Green	London
P4	Nut	Red	Oslo
P5	Screw	Red	Paris
P6	Screw	Red	Paris

P#	PNAME	COLOUR	CITY
P1	Bolt [1:1]	Blue [1:2]	London [1:3]
P2	Cam [2:2]	Green [3:3]	Oslo [4:4]
P3	Cog [3:3]	Red [4:6]	Paris [5:6]
P4	Nut [4:4]		
P5	Screw [5:6]		
P6			

Figure 19. Basic and Advanced versions of a Field Values Table, after Date [2004]

P#	PNAME	COLOUR	CITY
4	3	2	1
1	1	4	4
5	6	5	6
6	4	1	3
2	2	3	2
3	5	6	5

Figure 20 Record Reconstruction Table

•**The Zig Zag Algorithm**. This algorithm is used to rebuild the records. In this case it was used against the basic FVT version. As it is, it cannot handle the advanced version (condensed version).

"Step l: Go to cell [1, 1] of the Field Values Table and fetch the value stored there: namely, the part number P1. That value is the first field value (that is the P# field value) within a certain part record in the parts file.

Step 2: Go to the same cell (that is, cell [1, 1]) of the Record Reconstruction Table and fetch the value stored there: namely, the row number 4. That row number is interpreted to mean that the next field value (which is to say, the second or PNAME value) within the part record whose P# field value is P1 is to be found in the PNAME position of the

fifth row of the Field Values Table -in other words, in cell (4,2) of the Field Values Table. Go to that cell and fetch the value stored there (part name Nut).
Step 3: Go to the corresponding Record Reconstruction Table cell [4, 2] and fetch the row number stored there (4). The next (fourth or COLOUR) field value within the part record we are reconstructing is in the COLOUR position in the fourth row of the Field Values Table-in other words, in cell [4, 3]. Go to that cell and fetch the value stored there (colour Red).
Step 4: Go to the corresponding Record Reconstruction Table cell [4, 3] and fetch the value stored there (which is 1). The next (fourth or CITY) field value within the part record we are reconstructing is in the CITY position in the first row of the Field Values Table - in other words, in cell [1, 4]. Go to that cell and fetch the value stored there (city name London).
Step 5: Go to the corresponding Record Reconstruction Table cell [1, 4] and fetch the value stored there (1). Now, the "next" field value within the supplier record we are reconstructing looks like it ought to be the fifth such value; however, part records have only four fields, so that "fifth" wraps around to become the first. Thus, the "next" (first or P#) field value within the part record we are reconstructing is in the P# position in the first row of the Field Values Table - in other words, in cell [1, 1]. But that is where we came in, and the process stops."

3.5.3 Model Instantiation

The Transrelational model is new and no instantiation exists for research or commercial purposes; as an important part of the research was to benchmark the selected models, it was necessary to implement the essential algorithms of the storage part for the Transrelational model. This novel implementation of the storage mechanism for the Transrelational model is presented in Chapter 5.

3.6 Non Normal Form Databases (Non-1NF Databases)

In the decade of 1980 a line of research was investigated regarding a way to manage hierarchies into the Relational model. In 1988 the paper published by Roth, Korth and Silberschatz [Roth, 1988] has been considered as the milestone for the Non-First Normal Form databases.

In that paper the *Nest* and *Unnest* operators were defined to extend the relational model. According to Date [2006] the extensions were unnecessary; as the pure relational model provides the (useful) functionality that the "non-first-normal-form" model provides.
The operators introduced by the non-1NF allow the storage of a hierarchy or set (*Nest* operator) while the *Unnest* operator is used to retrieve the hierarchy.
In other words Non-1NF databases allow relational attributes to store sets instead of just one value. Figure 21 is an example of a Non-1NF table, it represents Students' Courses.

S_NAME	Course	
	C_NAME	GRADE
Jones	Math	A
	Science	B
Smith	Math	A
	Physics	C
	Science	A

Figure 21 Student Instance ($_{S1}$). From [Roth, 1988]

As can be observed a student has several courses and grades, and in relational terms the attribute Course is a set. This table does not conform to the 1st normal form definition.
If a second instance (S_2) of *Student* is considered

S_NAME	Course	
	C_NAME	GRADE
Jones	Physics	B
Smith	Chemestry	A
	English	B

Figure 22 Second Instance of *Student* (S_2). From [Roth, 1988]

Then the *Union* (U) of both instances is as follows:

S_NAME	Course	
	C_NAME	GRADE
Jones	Math	A
	Science	B
Jones	Physics	B
Smith	Math	A
	Physics	C
	Science	A
Smith	Chemestry	A
	English	B

Figure 23 Union of instances (S_1 U S_2). From [Roth, 1988]

Of course a better intuitive representation should be:

S_NAME	Course	
	C_NAME	GRADE
Jones	Math	A
	Science	B
	Physics	B
Smith	Math	A
	Physics	C
	Science	A
	Chemestry	A
	English	B

Figure 24 A better representation for Students Relations. From [Roth, 1988]

The main differences between the work of Roth *et al* and the work made on this thesis are:

The first consideration is that Roth et al works is managing information horizontally, while the work in this thesis considers the vertical management of data. Figure 25 shows the hierarchy of a Student represented horizontally.

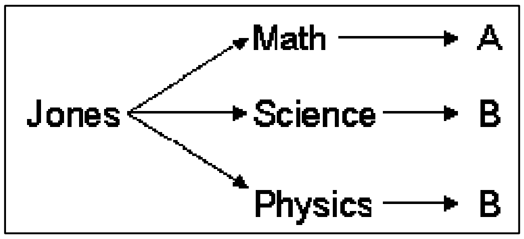


Figure 25 A Student hierarchy

In other words it represents a record with the consideration that its fields can be sets (see Figure 26)

Jones	Math	A
	Science	B
	Physics	B

Figure 26 A Non-1NF Record

In contrast the vertical approach manages everything as columns. Other fundamental difference regards the *linkage* information which ties the fields together [Date, 2004]. In the case of the Non-1NF the fields of a record are tight together by the physical contiguity while in the considered vertical alternative models the linkage structures are kept separated in other storage structures by doing this the values achieve independence of the physical contiguity.

Another difference which becomes important is the duplicates elimination which is fundamental in order to increase data density and decrease the size of the data warehouse. Analysing Figure 23 and Figure 24, the *union* of both instances have duplicates, and when applying the *Nest* operator duplicates are eliminated on the S_NAME (see Figure 24) but duplicates remain in the other fields (C_NAME and GRADE), in contrast on the Binary-Relational model, for example, duplicates are eliminated on all columns an example is represented on Figure 27. The composite attribute COURSE can not be represented on the Binary-Relational model, neither on the Transrelational model (which has a similar behaviour on its FVT) as both models consider that the original relations conform to the 1NF.

S_NAME	C_NAME	GRADE
Jones	Math	A
Smith	Science	B
	Physics	C
	Chemestry	
	English	

Figure 27 Binary-Relational representation

As can be seen in Figure 27, the binary-Relational eliminates duplicates in all columns and further compression can be achieved.

The only possible similar behaviour could be the interpretation of the Binary-Relational when relating for example that Math is used by Record 1 (Jones) and Record 2 (Smith) but that is not the way data is stored on the Binary-Relational instantiation, it uses the Binary Association Tables (BAT) and OIDs to keep track which value is utilised by which records, refer to section 3.4.2. This rough interpretation can be represented graphically as a series of Binary-Relations as shown in Figure 28, but this is not a valid Binary relation.

Jones	Math, Science, Physics
Smith	Math, Science, Physics, Chemestry, English

Jones	A,B
Smith	A,B,C

Math	Jones, Smith
Science	Jones, Smith
Physics	Jones, Smith
Chemestry	Smith
English	Smith

Math	A
Science	A,B
Physics	B,C
Chemestry	A
English	B,C

A	Jones, Smith
B	Jones, Smith
C	Smith

A	Math, Science, Chemestry
B	Science, Physiscs, English
C	Physics

Figure 28 Possible Binary-Relations interpretation.

The Non-First Normal Form databases were designed to manage records which fields need to store set values, and they stored data in a different fashion as the considered alternative models. Finally the Non-1NF Databases store data horizontally and do not eliminate duplicates at the column level. It still has the assumption that the fields of a record are linked by its physical contiguity. For these reasons this approach has not been considered within the alternative data models of this research, as it still keeping data horizontally and does not reduces data sparsity.

3.7 Conclusions

Some alternative data models have been selected to investigate their behaviour in Data Warehouse environments. The models were the Binary-Relational, the Associative/Triple Store and the Transrelational. These models have in common their approach to abandon the record structure, and manage data in a vertical way and eliminate duplicates at the column level.

They also eliminate nulls and by this they will reduce the data sparsity problem which is common in Data Warehouse environments. The chosen alternative data models will reduce sparsity and increase data density in the Data Warehouse. If it is true, then they could be a good alternative to be used within Data Warehouse environments.
Some of these models have existed for many years but they have been studied in isolation of each other and more oriented towards transactional systems, not for Data Warehouse environments, which is one of the novelty points of this thesis and the research carried out. The Associative/Triple Store has been studied in grater depth within the context of functional languages where it has succeeded.

Each data model stores data in a different way; In Table 2 the storage and linkage structures of each model are summarised.

Table 2 Alternative Models Structures

Model	**Storage Structure**	**Linkage Structure**
Relational	Table (Relation)	By position
Binary-Relational	Binary Table	Binary Association Tables and Joins
Associative	Items	Links
Transrelational	Field Values Table	Record Reconstruction table
Triple Store	Name Store	Triple Store

CHAPTER 4
Benchmarking Standards

This chapter aims to present some of the existing benchmarking standards in order to carry out the evaluation of the selected alternative data models (Binary-Relational, Triple-Store, Associative and Transrelational) in a Data Warehouse Environment. It was necessary to select a benchmark that can be considered useful to measure different models: well defined, impartial, complete and generally accepted in the research and commercial communities.

One of the major problems when comparing products, technology or data models is the lack of a benchmark that allows multiple products to be compared in a neutral way [Pendse, 2005]. Unfortunately, in the Data Warehouse / OLAP area, there is no benchmark that can be considered:

- Well defined in terms of the rules to be followed
- Impartial
- Complete
- Generally accepted in both research and industry

Some vendors have published private benchmarks [Pense, 2005], which lack credibility because high numbers are published but the benchmarking methodology is not described in detail [Pendse, 2005]; such benchmarks frequently are frequently used to advertise specific products. Some of these benchmarks are not properly defined and cannot be generally accepted. For example, Microstrategy in 1997 claimed that “Stress test enables over 40,000 users to access Data Warehouse via the Web” [Pendse, 2005], but the methodology has never been published, neither the conditions nor the kind of tests run. Such kinds of “benchmarks” are generally designed to highlight a product’s strengths and bypass their weaknesses. Another example of this kind of benchmark is “The Drill Down benchmark” [Boncz, 1998] which considers only a particular environment, so it cannot be considered a broadly accepted benchmark. This benchmark was defined for Boncz *et al*. to highlight the architecture of their own MonetDB DBMS [MonetDB, 2004a].

In the following sections, the characteristics of some of the principal Data Warehouse/OLAP benchmarks are analysed.

OLAP Council's APB-1

In 1995, an initial effort to have a standard Data Warehouse/OLAP benchmark was made, when 18 OLAP vendors created the OLAP Council with the purpose of defining an industrial benchmark in order to compare OLAP performance [Pendse, 2005]. In 1996, the OLAP Council published the APB-1 benchmark (Analytical Processing Benchmark Version 1).

The APB-1 benchmark had some deficiencies:

- It was not designed towards any particular product; it seems to be more oriented towards Essbase and Express multidimensional databases [Pendse, 2005].
- APB-1 was designed for MOLAP servers; none of the OLAP Council members was a ROLAP vendor.
- No standards for auditing were defined.

Much interpretation was left open and finally this concluded in a lack of credibility for the published results.

In 1998, a modified version of the APB-1 benchmark was published. It had tighter rules: for example forcing full disclosure and forcing vendors to perform the full set of tests and not just those that favour their own products, which is good to have for fair conditions on the benchmarks. However, the modified version of the APB-1 benchmark did not achieve industrial acceptance. Cognos, Microsoft and Microstrategy, which are major vendors in the Data Warehouse / OLAP market, have never shown any interest in it [Pendse, 2005].

The APB-1 has reasonably complex calculations but with very small volumes; none of the published runs included more than 5GB. Consequently, the total input data set can be held in RAM and query performance could be favourable. Corporate Data Warehouses can easily exceed 5GB and reach Terabyte sized tables; therefore APB-1 is not relevant for this current research.

APB-1 rules allow vendors to change numerous parameters. Tests are not comparable. Also, APB-1 does not force vendors to publish the size/cost of the hardware used in tests, thus making it unusable. APB-1 lacks wide acceptance and has been declining in the last few years [Pendse, 2005]

APB-1 does not address the data density or the data sparsity aspects (research objectives 1 and 2). Therefore the APB-1 benchmark is not suitable for use in this research, and more general-purpose benchmarks that include ROLAP characteristics are needed.

Microsoft 1TB benchmark

In 1999, Microsoft published an OLAP benchmark able to manage 1TB of raw data. Its purpose was to demonstrate that the combination of SQL Server and OLAP services products were able to scale to large data volumes [Pendse, 2005].

What was done was to use the capabilities of the *OLAP services* product to manage ROLAP aggregates that were manually computed and stored in an SQL Server. The queries were studied and some specific aggregated tables were pre-built to benefit query times [Pendse, 2005].

It should be noticed that Microsoft loaded 1TB of raw data into the SQL Server, but once loaded, pre-computed and storing all necessary aggregates, the Data Warehouse size was 2.6 TB [Pendse, 2005]. This is an example of the database explosion phenomenon which is related to research objectives 1, 2 and 6 of this thesis.

TPC Benchmarks

The Transaction Processing Council (TPC) has a suite of benchmarks, each targeted to different computing environments. The TPC Council was founded in 1988, and it has around 40 to 55 members for the last several years [TPCH, 2002].

As there are not well-accepted and established benchmarks for Data Warehouse environments, the TPC benchmarks were investigated. TPC benchmarks are preferred in this thesis, because they are well accepted in both industry and in the research community.

TPC-A, TPC-B and TPC-C

TPC-A, TPC-B and TPC-C were the initial TPC benchmarks oriented to OLTP systems. Now they are obsolete and have been replaced by the TPC-C benchmark. These benchmarks will not be used in this research because they were designed to test transactional systems (OLTP) rather than Data Warehouse/OLAP systems.

TPC-D, TPC-H and TPC-R

TCP-D Benchmark was the initial Decision Support Benchmark. Now it is obsolete but has evolved into two different benchmarks, TPC-H and TPC-R, which are targeted to Decision Support Systems that are typical applications of a Data Warehousing Environment.

The TPC-H benchmark was chosen for this research because it offered the best conditions to evaluate *pristine* Data Models and not technology. The term *pristine* has been used in this research to examine the data models at their base level, unencumbered by the additional tools that have been developed to improve their performance. The only type of auxiliary structures allowed in TPC-H are indexes on primary and foreign keys (it should be remembered that indexes are not actually part of the Relational model [Codd, 1990]), which makes TPC-H more restrictive and assumes no previous knowledge of the queries to be executed against the Data Warehouse.

In contrast, TPC-R allows the use of extended Relational technology, such as indexes over any column, *join indexes, materialized views, pre-aggregates computation,* and practically any technology that the DBMS can have to improve performance and it was preferred to run the laboratory test without any database specific optimisation.
The TPC-H version used in this research was Version 2.1.0 of the benchmark [TPCH, 2002].
The research scenario involves the testing of several data models and products, and the avoidance of biased results was a priority for this research. Research results could be biased; for example, having more knowledge of a specific product/model can favour such DBMS or models, or as more experience is obtained while running the benchmark the results may be biased as well, because for the last tested data model more experience on the benchmark would be acquired. Therefore, it was decided to use TPC-H because it is a better benchmark that will allow the measuring of Data Models instead of measuring the technology features of the DBMSs.

TPC-H has been designed to run on Relational based products, which must be commercially available. It consists of 22 queries that must be executed via an interface using dynamic SQL [TPCH, 2002] and must be run against a database consisting of eight tables, the logical schema of which is presented in section 4.2. The twenty two queries of the TPC-H are listed as they were run during the experiments of this thesis in appendix A.4 accompanied by a business explanation.

4.1 Acceptance of the TPC-H benchmark

As mentioned before, one of the criteria for choosing a benchmark is its acceptance in both research and industry. TPC-H satisfies this condition as it is widely accepted in both communities.

4.1.1 Acceptance in research

TPC-H and its ancestor TPC-D have been extensively used in research on Data Warehouse optimization [Ramamurthy, 2002], Query Optimization [Chen, 2001], Data Mining, Data Models [Boncz, 2002], Using Compression in Databases [Westman, 2000].

4.1.2 Acceptance in Industry

TPC-H is widely accepted in industry. In Table 3 the members of the TPC council are listed as of January 1998 according to TPC-H [2002], and are the members listed on the TPC-H 2.1.0 definition used in this research. The relevance of mentioning TPC-H members is because they are the major vendors for Data Warehousing.

Table 3 Members of the TPC Council

Adaptec	Information Decisions	Samsung Electronics Corp.
Advanced Logic Research	Informix Software	Sanford Bernstein
Amdahl	Intel Corporation	Sarion Systems Research
BEA Systems	Intergraph	SCO
Bull S.A.	ITOM International	Sequent Computer
Compaq Computer Corp.	Microsoft Corporation	Siemens Nixdorf
Computer Associates	Mitsubishi Electric Corp.	Information Systems
Data General Corporation	Motorola	Silicon Graphics
Debis Systems	Morgan Stanley	Software AG
Dell Computer Corporation	Mylex	Stratus Computer
Digital Equipment Corp.	NCR	Sun Microsystems
EDS	NEC Systems Laboratory	Sybase
EMC Corporation	Nikkei Business Publications	Tandem Computers
Fujitsu/ICL	Novell	Toshiba Corporation
Hewlett-Packard	OKI Electric Industry	Tricord Systems Inc.
Hitachi SW	Olivetti S.p.A	Unisys Corporation
IBM Corporation	Oracle Corporation	University of Pavia
IDEAS International	Performance Tuning Corp.	White Cross Systems

4.2 TPC-H Database Schema

The TPC-H schema consists of eight entities, shown in Figure 29. It has been designed in order to benchmark Decision Support Systems. It models a generic business that sells products, produces invoices, and has suppliers that provide items to be sold to the clients.

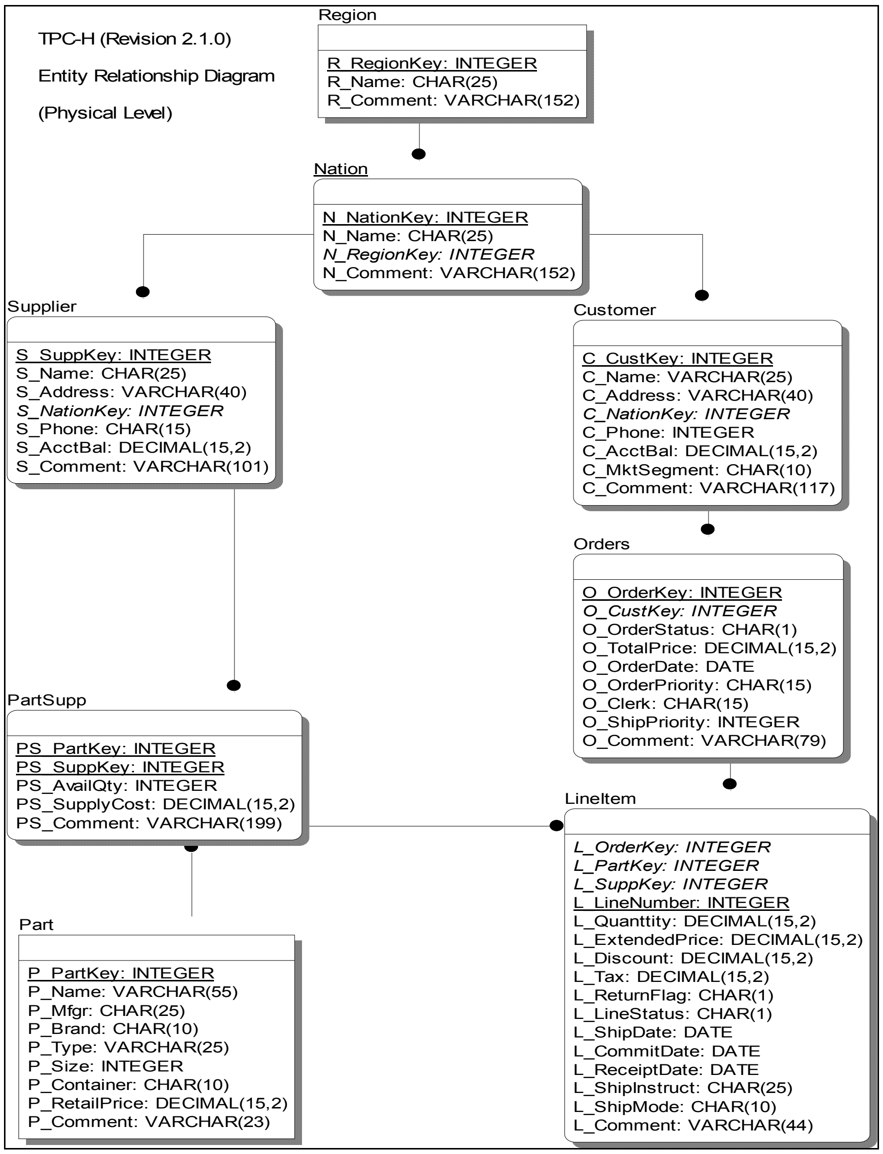


Figure 29 TPC-H Database ER Diagram

4.2.1 TPC-H Schema Usage

In Table 4 the columns used by each query has been represented, as can be seen queries used all tables, but some columns are never used (R_comment, N_comment, P_retailprice, P_comment, PS_comment). The main part of the queries are solved using numeric attributes, and in few times the “comments” columns are used, but those “comment” columns occupy large portion of the database size. This fact is more relevant for the n-ary-Relational model because of its horizontal management of data, it need to retrieve the complete row, by contrast the Binary-Relational model, which manages everything as independent columns does not have the need of reading those large columns which are irrelevant for the queries resolution. This approach makes Binary-relational able to save large amounts of I/O operations and this will be reflected on a better performance for the Binary-Relational model (see Chapter 6 and Chapter 7).

Table 4 TPC-H Columns Usage by Query

Query -->	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22
Column																						
R_REGIONKEY		■			■			■														
R_NAME		■			■			■														
R_COMMENT																						
N_NATIONKEY		■			■		■	■	■	■	■										■	
N_NAME		■					■	■	■	■	■										■	
N_REGIONKEY		■			■			■														
N_COMMENT																						
S_SUPPKEY		■			■		■	■	■		■				■	■				■	■	
S_NAME		■																		■	■	
S_ADDRESS		■																		■		
S_NATIONKEY		■			■		■	■	■		■										■	
S_PHONE		■																				
S_ACCTBAL		■																				
S_COMMENT		■														■						
C_CUSTKEY			■		■		■	■		■			■					■				■
C_NAME										■								■				
C_ADDRESS										■												
C_NATIONKEY					■		■	■		■												
C_PHONE										■												■
C_ACCTBAL										■												■
C_MKTSEGMENT			■																			
C_COMMENT										■			■									
P_PARTKEY		■						■	■					■		■	■		■	■		
P_NAME									■											■		
P_MFGR		■																				
P_BRAND																■	■		■			
P_TYPE		■						■						■		■						
P_SIZE		■														■			■			
P_CONTAINER																	■		■			
P_RETAILPRICE																						
P_COMMENT																						
PS_PARTKEY		■							■		■					■				■		
PS_SUPPKEY		■							■		■					■				■		
PS_AVAILQTY											■									■		
PS_SUPPLYCOST		■							■		■											
PS_COMMENT																						
O_ORDERKEY			■	■	■		■	■	■	■		■	■					■			■	■
O_CUSTKEY			■		■		■	■		■			■					■				■
O_ORDERSTATUS																					■	■
O_TOTALPRICE																		■				■
O_ORDERDATE			■	■	■			■	■	■								■				■
O_ORDERPRIORITY			■	■								■										■
O_CLERK																						■
O_SHIPPRIORITY			■																			■
O_COMMENT																						■
L_ORDERKEY			■	■	■		■	■	■	■		■						■			■	
L_PARTKEY				■				■	■					■			■		■	■	■	
L_SUPPKEY				■	■		■	■	■						■					■	■	
L_LINENUMBER				■																	■	
L_QUANTITY	■			■		■			■								■	■	■	■	■	
L_EXTENDEDPRICE	■		■	■	■	■	■	■	■	■				■	■		■		■		■	
L_DISCOUNT	■		■	■	■	■		■	■	■				■	■				■		■	
L_TAX	■			■																	■	
L_RETURNFLAG	■			■						■											■	
L_LINESTATUS	■			■																	■	
L_SHIPDATE	■		■	■		■	■					■		■	■					■	■	
L_COMMITDATE				■								■									■	
L_RECEIPTDATE				■								■									■	
L_SHIPINSTRUCT				■															■		■	
L_SHIPMODE				■								■							■		■	
L_COMMENT				■																	■	

4.3 TPC-H General Characteristics

From the eight entities of the TPC-H model, the two largest are *Lineitem* which can be considered as the Fact Table of a Data Warehouse, because it contains the detailed data of the model. The second largest table is *Orders*. The main part of the Data Warehouse size is kept between these two tables.

Especial attention was given to these two tables during the experimental phase of the research, as achieving good management of them will have a global effect on the main aspects of the research: increase data density (objective 1) and better handling of data sparsity (objective 2).

The other tables can be considered "dimensions" of a Data Warehouse schema and are small compared with Orders and Lineitem.

The total schema consists of 61 attributes and 42% of these are string values (26 attributes). According to Chen [2001], there has not been much work in compressing string values. The approach followed in this thesis does not discriminate between string or numeric values, because the research regards the use of an alternative data model (objective 1) which eliminates duplicate data at the column level by itself and does not rely on compression algorithms as others have done [Balakrishna, 1994; Chen, 2001; Morri, 2002; Poss, 2003].

The TPC-H definition allows indexing over only those columns that participate as primary or foreign keys [TPCH, 2002]. The TPC council provides programs and utilities to set up a TPC-H environment. Those can be downloaded from the TPC site [TPC, 2009].

- First the proper Data Definition Language (DDL) statements should be chosen according to the target RDBMS; in this research it was Oracle DBMS.
- Second the *qgen* program generates the queries with the syntax for the target RDBMS.
- The third step is to run the *dbgen* program that generates data to populate the Database.

Once the TPC-H database is populated, queries can be run and measurements can be recorded.

4.4 TPC-H Queries Analysis

In this section the TPC-H queries are presented as they are defined in the standard [TPCH, 2002], and they have been taken from the same source [TPCH, 2002]. It is important to mention that some queries were modified in order to use standard ANSI SQL and this way made them run on more data models. The main change was that, instead of using a derived table, views were created first and then the select statement was executed. The queries that were modified are: Q7, Q8, Q9, Q13 and Q22, according to the rules allowed in TPCH [2002]. In order to identify the original queries, the modified ones have been renamed as Q77, Q88, Q99, Q1313 and Q2222. Both the original and the modified Queries have been included. The Queries Analysis has been based on the work of Ballinger [2009]. That work was made to analyse the TPC-D benchmark queries shapes and behaviour. In this thesis the work of Ballinger has been updated for the TPC-H benchmark. The main differences are:

TPC-D consisted on 17 queries while TPC-H consists of 22 queries.

- There is an error on Ballinger functionality analysis of Query 6, as he said 5-table join while it should be 6-table join.
- Queries 8 and 9 have changed from TPC-D to TPC-H, now both queries are solved by using sub-queries, but the behaviour and results are similar.
- Query 13 has been changed in TPC-D it was called “Sales Clerk Performance” and now in TPC-H it is called “Customer Distribution” and the shapes of the queries are completely different.
- In [Ballinger, 2009] Query 16 is described but it is not analysed in contrast Query 17 is analysed but never described. Both queries have been fully described and analysed in this thesis.
- The analysis of Queries 18 to 22 has been made in this thesis, as there did not exist before in TPC-D.

Q1. Pricing Summary Report Query

This query reports the amount of business that was billed, shipped and returned. The Pricing Summary Report Query provides a summary pricing report for all lineitems shipped as of a given date, which is within 60 and 120 days of the greatest ship date contained in the database.

```
-- @(#)1.sql  2.1.8.1
-- TPC-H/TPC-R Pricing Summary Report Query (Q1)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
set echo on
spool salida1.lis

select
       l_returnflag,
       l_linestatus,
       sum(l_quantity) as sum_qty,
       sum(l_extendedprice) as sum_base_price,
       sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
       sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
       avg(l_quantity) as avg_qty,
       avg(l_extendedprice) as avg_price,
       avg(l_discount) as avg_disc,
       count(*) as count_order
from
       lineitem
where
       l_shipdate <= date '1998-12-01' - interval '92' day (3)
group by
       l_returnflag,
       l_linestatus
order by
       l_returnflag,
       l_linestatus;

exit
```

Analysis: Query 1 performs multiple aggregations and summaries by reading and processing over 95% of the rows of the database's largest table. Only this single table is scanned with a very low number of rows being returned.

Q2. Minimum Cost Supplier Query

The Minimum Cost Supplier Query finds for each part of a certain type and size, the supplier in a given region who can supply it at minimum cost. If several suppliers in that region offer the desired part type and size at the same cost, the query lists the parts from suppliers with the 100 highest account balances.

```
-- @(#)2.sql  2.1.8.2
-- TPC-H/TPC-R Minimum Cost Supplier Query (Q2)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
set echo on
spool salida2.lis

select
       s_acctbal,s_name,
       n_name,p_partkey,
       p_mfgr,s_address,
       s_phone,s_comment
from
       part, supplier, partsupp, nation, region
where
       p_partkey = ps_partkey
       and s_suppkey = ps_suppkey
       and p_size = 22
       and p_type like '%TIN'
       and s_nationkey = n_nationkey
       and n_regionkey = r_regionkey
       and r_name = 'AFRICA'
       and ps_supplycost = (
              select
                     min(ps_supplycost)
              from
                     partsupp, supplier, nation, region
              where
                     p_partkey = ps_partkey
                     and s_suppkey = ps_suppkey
                     and s_nationkey = n_nationkey
                     and n_regionkey = r_regionkey
                     and r_name = 'AFRICA'
       )
order by
       s_acctbal desc,
       n_name,
       s_name,
       p_partkey;
exit
```

Analysis: Query 2 is a correlated sub-query based on a 5-table join in both outer query and inner query. Close to 5% of the Supplier rows result from the selection criteria and query processing, but only the 100 Suppliers with the highest account balances are returned.

Q3. Shipping Priority Query

Query 3 retrieves the shipping priority and potential revenue Orders which have the largest revenue among those that had not been shipped as up to a given date. Orders are listed in decreasing order of revenue. This query returns the top 10 unshipped orders according to their potential revenue.

```
-- @(#)3.sql 2.1.8.1
-- TPC-H/TPC-R Shipping Priority Query (Q3)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
set echo on
spool salida3.lis

select
       l_orderkey,
       sum(l_extendedprice * (1 - l_discount)) as revenue,
       o_orderdate,
       o_shippriority
from
       customer,
       orders,
       lineitem
where
       c_mktsegment = 'AUTOMOBILE'
       and c_custkey = o_custkey
       and l_orderkey = o_orderkey
       and o_orderdate < date '1995-03-01'
       and l_shipdate > date '1995-03-01'
group by
       l_orderkey,
       o_orderdate,
       o_shippriority
order by
       revenue desc,
       o_orderdate;
exit
```

Analysis: Query 3 performs a 3-table join on three of the larger tables in the database. Anywhere from 1/5 to _ of the rows of each of the 3 tables participate in the joins, with a final aggregation that produces a very high number of rows, in the millions for most volume points. Only the 10 Orders with the highest revenue are returned.

Q4. Order Priority Checking Query

This query determines how well the order priority system is working and gives an assessment of customer satisfaction.

The Order Priority Checking Query counts the number of orders ordered in a given quarter of a given year in which at least one lineitem was received by the customer later than its committed date. The query lists the count of such orders for each order priority sorted in ascending priority order.

```
-- @(#)4.sql 2.1.8.1
-- TPC-H/TPC-R Order Priority Checking Query (Q4)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
set echo on
spool salida4.lis

select
       o_orderpriority,
       count(*) as order_count
from
       orders
where
       o_orderdate >= date '1993-03-01'
       and o_orderdate < date '1993-03-01' + interval '3' month
       and exists (
              select
                     *
              from
                     lineitem
              where
                     l_orderkey = o_orderkey
                     and l_commitdate < l_receiptdate
       )
group by
       o_orderpriority
order by
       o_orderpriority;
exit
```

Analisys: Query 4 is a correlated sub-query in which about 1/28 of the Lineitem rows are selected for evaluation based on order date. An aggregation that produces a count by priority produces 5 rows in the answer set.

Q5. Local Supplier Volume Query

This query lists the revenue volume done through local suppliers.

The Local Supplier Volume Query lists for each nation in a region the revenue volume that resulted from lineitem transactions in which the customer ordering parts and the supplier filling them were both within that nation. The query is run in order to determine whether to institute local distribution centres in a given region. The query considers only parts ordered in a given year. The query displays the nations and revenue volume in descending order by revenue. Revenue volume for all qualifying lineitems in a particular nation is defined as sum(l_extendedprice * (1 -l_discount)).

```
-- @(#)5.sql  2.1.8.1
-- TPC-H/TPC-R Local Supplier Volume Query (Q5)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
set echo on
spool salida.lis

-- query 5

select
       n_name,sum(l_extendedprice * (1 - l_discount)) as revenue
from
       customer,orders,lineitem,
       supplier,nation,region
where
       c_custkey = o_custkey
       and l_orderkey = o_orderkey
       and l_suppkey = s_suppkey
       and c_nationkey = s_nationkey
       and s_nationkey = n_nationkey
       and n_regionkey = r_regionkey
       and r_name = 'AMERICA'
       and o_orderdate >= date '1997-01-01'
       and o_orderdate < date '1997-01-01' + interval '1' year
group by
       n_name
order by
       revenue desc;
exit
```

Analisys: Query 5, this is a 6-table join of large and small tables, where the data aggregated is reduced down to 1/5 of the Customers and Suppliers (representing one Region out of five) and 1/7 of the Lineitems (one year out of seven). The largest detail table has no direct selection applied to it. Five rows are returned, constituting the revenue for each nation in the selected region.

Q6. Forecasting Revenue Change Query

This query quantifies the amount of revenue increase that would have resulted from eliminating certain companywide discounts in a given percentage range in a given year. Asking this type of "what if" query can be used to look for ways to increase revenues.

The Forecasting Revenue Change Query considers all the lineitems shipped in a given year with discounts between DISCOUNT-0.01 and DISCOUNT+0.01. The query lists the amount by which the total revenue would have increased if these discounts had been eliminated for lineitems with l_quantity less than quantity. Note that the potential revenue increase is equal to the sum of [l_extendedprice * l_discount] for all lineitems with discounts and quantities in the qualifying range.

```
-- @(#)6.sql  2.1.8.1
-- TPC-H/TPC-R Forecasting Revenue Change Query (Q6)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
set echo on
spool salida6.lis

select
       sum(l_extendedprice * l_discount) as revenue
from
       lineitem
where
       l_shipdate >= date '1997-01-01'
       and l_shipdate < date '1997-01-01' + interval '1' year
       and l_discount between 0.03 - 0.01 and 0.03 + 0.01
       and l_quantity < 25;
exit
```

Analisys: Query 6 accesses the large detail table only (Lineitem) selecting about 12% of the rows, and returning a single column answer.

Q7. Volume Shipping Query

This query determines the value of goods shipped between certain nations to help in the re-negotiation of shipping contracts.

The Volume Shipping Query finds, for two given nations, the gross discounted revenues derived from lineitems in which parts were shipped from a supplier in either nation to a customer in the other nation during 1995 and 1996. The query lists the supplier nation, the customer nation, the year, and the revenue from shipments that took place in that year. The query orders the answer by Supplier nation, Customer nation, and year (all ascending).

```
-- @(#)7.sql  2.1.8.1 TPC-H/TPC-R Volume Shipping Query (Q7)
-- Functional Query Definition. Approved February 1998
-- using 472029144 as a seed to the RNG
set echo on;spool salida7.lis;
select
       supp_nation, cust_nation, l_year,
       sum(volume) as revenue
from
       (
       select
              n1.n_name as supp_nation,
              n2.n_name as cust_nation,
              extract(year from l_shipdate) as l_year,
              l_extendedprice * (1 - l_discount) as volume
       from
              supplier,
              lineitem,
              orders,
              customer,
              nation n1,
              nation n2
       where
              s_suppkey = l_suppkey
              and o_orderkey = l_orderkey
              and c_custkey = o_custkey
              and s_nationkey = n1.n_nationkey
              and c_nationkey = n2.n_nationkey
              and ((n1.n_name = 'EGYPT' and n2.n_name = 'UNITED STATES')
              or (n1.n_name = 'UNITED STATES' and n2.n_name ='EGYPT'))
              and l_shipdate between date '1995-01-01' and date'1996-12-31')
group by
       supp_nation,cust_nation,l_year
order by
       supp_nation,cust_nation,l_year;
exit
```

Analysis: Query 7 is a 6-table join that requires a small 25-row table, NATION, to be aliased and processed as though it were two distinct look-up tables. A date constraint selects 2/7 of the Lineitem rows, while a join to the lookup tables result in 2/5 of the Customers and Suppliers being selected. Four rows are returned.

Q77. Volume Shipping Query (modified version of Q7)

```
-- using 472029144 as a seed to the RNG
alter session set NLS_DATE_FORMAT='YYYY-MM-DD';
set echo on
set feed on
create view q7 as (
       select
              n1.n_name as supp_nation,
              n2.n_name as cust_nation,
              extract(year from l_shipdate) as l_year,
              l_extendedprice * (1 - l_discount) as volume
       from
              supplier,
              lineitem,
              orders,
              customer,
              nation n1,
              nation n2
       where
              s_suppkey = l_suppkey
              and o_orderkey = l_orderkey
              and c_custkey = o_custkey
              and s_nationkey = n1.n_nationkey
              and c_nationkey = n2.n_nationkey
              and ((n1.n_name = 'UNITED STATES' and n2.n_name = 'EGYPT')
              or (n1.n_name = 'EGYPT' and n2.n_name = 'UNITED STATES'))
              and l_shipdate between date '1995-01-01' and date '1996-12-31'
        );
-- spool query77.lst;
select supp_nation,
       cust_nation,
       l_year,
       sum(volume) as revenue
from q7
group by
    supp_nation,
    cust_nation,
    l_year
order by
    supp_nation,
    cust_nation,
    l_year;
drop view q7;
exit
```

Q8. National Market Share Query

This query determines how the market share of a given nation within a given region has changed over two years for a given part type.

The market share for a given nation within a given region is defined as the fraction of the revenue, the sum of [l_extendedprice * (1-l_discount)], from the products of a specified type in that region that was supplied by suppliers from the given nation. The query determines this for the years 1995 and 1996 presented in this order.

```
-- @(#)8.sql 2.1.8.1
-- TPC-H/TPC-R National Market Share Query (Q8)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
set echo on
spool salida8.lis

select
       o_year,
       sum(case
              when nation = 'UNITED STATES' then volume
              else 0
       end) / sum(volume) as mkt_share
from
       (select
              extract(year from o_orderdate) as o_year,
              l_extendedprice * (1 - l_discount) as volume,
              n2.n_name as nation
       from
              part,supplier,
              lineitem,orders,
              customer,nation n1,
              nation n2,region
       where
              p_partkey = l_partkey
              and s_suppkey = l_suppkey
              and l_orderkey = o_orderkey
              and o_custkey = c_custkey
              and c_nationkey = n1.n_nationkey
              and n1.n_regionkey = r_regionkey
              and r_name = 'AMERICA'
              and s_nationkey = n2.n_nationkey
             and o_orderdate between date '1995-01-01' and date '1996-12-31'
              and p_type = 'LARGE PLATED NICKEL'
       )
group by
       o_year
order by
       o_year;
exit
```

Analysis: Query 8 is a correlated sub-query based on an 8-table join including the NATION table twice. CASE (if/else logic) is used to determine and return a percent value based on two different years. Selection constraints qualify 1/50 of the Part rows, and 1/25 of the Nations.

Q88. National Market Share Query (modified version of Q8)

```
-- using 472029144 as a seed to the RNG
set echo on
set feed on
alter session set NLS_DATE_FORMAT='YYYY-MM-DD';
create view q8 as
       (
       select
              extract(year from o_orderdate) as o_year,
              l_extendedprice * (1 - l_discount) as volume,
              n2.n_name as nation
       from
              part,
              supplier,
              lineitem,
              orders,
              customer,
              nation n1,
              nation n2,
              region
       where
              p_partkey = l_partkey
              and s_suppkey = l_suppkey
              and l_orderkey = o_orderkey
              and o_custkey = c_custkey
              and c_nationkey = n1.n_nationkey
              and n1.n_regionkey = r_regionkey
              and r_name = 'AMERICA'
              and s_nationkey = n2.n_nationkey
             and o_orderdate between date '1995-01-01' and date '1996-12-31'
                     and p_type = 'LARGE PLATED NICKEL'
       );
-- spool query88.lst;
select
       o_year,
       sum(case
              when nation = 'UNITED STATES' then volume
              else 0
       end) / sum(volume) as mkt_share
from q8
group by
       o_year
order by
       o_year;
drop view q8;
exit
```

Q9. Product Type Profit Measure Query

This query determines how much profit is made on a given line of parts, broken down by supplier nation and year.

The Product Type Profit Measure Query finds, for each nation and each year, the profit for all parts ordered in that year that contain a specified substring in their names and that were filled by a supplier in that nation. The profit is defined as the sum of [(l_extendedprice*(1-l_discount)) - (ps_supplycost * l_quantity)] for all lineitems describing parts in the specified line. The query lists the nations in ascending alphabetical order and, for each nation, the year and profit in descending order by year (most recent first).

```
-- @(#)9.sql 2.1.8.1
-- TPC-H/TPC-R Product Type Profit Measure Query (Q9)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
set echo on
spool salida9.lis

select
      nation,o_year,sum(amount) as sum_profit
from
      (
      select
         n_name as nation,extract(year from o_orderdate) as o_year,
          l_extendedprice*(1-l_discount)-ps_supplycost*l_quantity as amount
      from
            part, supplier,lineitem, partsupp,orders, nation
      where
            s_suppkey = l_suppkey
            and ps_suppkey = l_suppkey
            and ps_partkey = l_partkey
            and p_partkey = l_partkey
            and o_orderkey = l_orderkey
            and s_nationkey = n_nationkey
            and p_name like '%mint%'
      )
group by
      nation,o_year
order by
      nation,o_year desc;
exit
```

Analysis: Query 9 is a correlated sub-query of a 6-table join with selection criteria applied to only one small table, Part. A text string search is done against the PART table, resulting in 5% of the Part rows being selected. Revenue is aggregated for each combination of Year and Nation, returning 175 rows.

Q99. Product Type Profit Measure Query (modified version of Q9)

```
-- using 472029144 as a seed to the RNG
set echo on
set feed on
alter session set NLS_DATE_FORMAT='YYYY-MM-DD';
create view q9 as
       (
       select
           n_name as nation,
           extract(year from o_orderdate) as o_year,
           l_extendedprice*(1-l_discount)-ps_supplycost*l_quantity as amount
       from
              part,
              supplier,
              lineitem,
              partsupp,
              orders,
              nation
       where
              s_suppkey = l_suppkey
              and ps_suppkey = l_suppkey
              and ps_partkey = l_partkey
              and p_partkey = l_partkey
              and o_orderkey = l_orderkey
              and s_nationkey = n_nationkey
              and p_name like '%mint%'
       );
-- spool query99.lst;
select
       nation,
       o_year,
       sum(amount) as sum_profit
from q9
group by
       nation,
       o_year
order by
       nation,
       o_year desc;
drop view q9;
exit
```

Q10. Returned Item Reporting Query

The query identifies customers who might be having problems with the parts that are shipped to them.

The Returned Item Reporting Query finds the top 20 customers, in terms of their effect on lost revenue for a given quarter, who have returned parts. The query considers only parts that were ordered in the specified quarter. The query lists the customer's name, address, nation, phone number, account balance, comment information and revenue lost. The customers are listed in descending order of lost revenue. Revenue lost is defined as sum(l_extendedprice*(1-l_discount)) for all qualifying lineitems.

Return the first 20 selected rows.

```
-- @(#)10.sql 2.1.8.1
-- TPC-H/TPC-R Returned Item Reporting Query (Q10)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
set echo on
spool salida10.lis

select
       c_custkey,c_name,
       sum(l_extendedprice * (1 - l_discount)) as revenue,
       c_acctbal,n_name,c_address,c_phone,c_comment
from
       customer, orders, lineitem, nation
where
       c_custkey = o_custkey
       and l_orderkey = o_orderkey
       and o_orderdate >= date '1993-10-01'
       and o_orderdate < date '1993-10-01' + interval '3' month
       and l_returnflag = 'R'
       and c_nationkey = n_nationkey
group by
       c_custkey, c_name, c_acctbal, c_phone,n_name, c_address, c_comment
order by
       revenue desc;
exit
```

Analysis: Query 10 is a 4-table join of three large tables and one look-up table. Customer detail is returned by this query, alongside of only one column of summarized data. 1/28 of the Order rows qualify based on date, while 1 in 4 Lineitems match the criteria of having a return flag of 'R'. Millions of rows are returned from this query, but the final sort determines which 20 rows are to be displayed in the answer set.

Q11. Important Stock Identification Query

This query finds the most important subset of suppliers' stock in a given nation.

The Important Stock Identification Query finds, from scanning the available stock of suppliers in a given nation, all the parts that represent a significant percentage of the total value of all available parts. The query displays the part number and the value of those parts in descending order of value.

```
-- @(#)11.sql 2.1.8.1
-- TPC-H/TPC-R Important Stock Identification Query (Q11)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
set echo on
spool salida11.lis
-- query 11

Select ps_partkey,
        sum(ps_supplycost * ps_availqty) as value
from
       partsupp,
       supplier,
       nation
where
       ps_suppkey = s_suppkey
       and s_nationkey = n_nationkey
       and n_name = 'JAPAN'
group by
       ps_partkey having
              sum(ps_supplycost * ps_availqty) > (
                     select
                            sum(ps_supplycost * ps_availqty) * 0.0001000000
                     from
                            partsupp,
                            supplier,
                            nation
                     where
                            ps_suppkey = s_suppkey
                            and s_nationkey = n_nationkey
                            and n_name = 'JAPAN'
              )
order by
       value desc;
exit
```

Analysis: Query 11 is a sub-query with both the inner and outer query composed of a 3-table join of the same 3 moderate and small tables. A HAVING clause associates the two sub-queries. 1/25 of these tables' rows are selected, with several million rows being returned in the final answer set.

Q12. Shipping Modes and Order Priority Query

This query determines whether selecting less expensive modes of shipping is negatively affecting the critical-priority orders by causing more parts to be received by customers after the committed date.

The Shipping Modes and Order Priority Query counts, by ship mode, for lineitems actually received by customers in a given year and the number of lineitems belonging to orders for which the l_receiptdate exceeds the l_commitdate for two different specified ship modes. Only lineitems that were actually shipped before the l_commitdate are considered. The late lineitems are partitioned into two groups: those with priority URGENT or HIGH, and those with a priority other than URGENT or HIGH.

```
-- @(#)12.sql 2.1.8.1
-- TPC-H/TPC-R Shipping Modes and Order Priority Query (Q12)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
-- query 12

select l_shipmode,
       sum(case
              when o_orderpriority = '1-URGENT'
                or o_orderpriority = '2-HIGH' then 1
              else 0
       end) as high_line_count,
       sum(case
              when o_orderpriority <> '1-URGENT'
                     and o_orderpriority <> '2-HIGH' then 1
              else 0
       end) as low_line_count
from
       orders, lineitem
where
       o_orderkey = l_orderkey
       and l_shipmode in ('REG AIR', 'FOB')
       and l_commitdate < l_receiptdate
       and l_shipdate < l_commitdate
       and l_receiptdate >= date '1997-01-01'
       and l_receiptdate < date '1997-01-01' + interval '1' year
group by
       l_shipmode
order by
       l_shipmode;
exit
```

Analysis: Query 12 is a two-table join of the two largest tables, with 1/7 of the Lineitems being selected based on date, and all Orders participating in the join. CASE functionality is used to allow a single processing of both tables in order to count the qualifying rows for two different sets of priority criteria. Two rows are returned from the aggregation.

Q13. Customer Distribution Query

This query seeks relationships between customers and the size of their orders.

This query determines the distribution of customers by the number of orders they have made, including customers who have no record of orders, past or present. It counts and reports how many customers have no orders, how many have 1, 2, 3, etc. A check is made to ensure that the orders counted do not fall into one of several special categories of orders. Special categories are identified in the order comment column by looking for a particular pattern.

```
-- @(#)13.sql 2.1.8.1
-- TPC-H/TPC-R Customer Distribution Query (Q13)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
set echo on
spool salida13.lis

-- query 13

select
       c_count,
       count(*) as custdist
from
       (
              select
                     c_custkey,
                     count(o_orderkey)
              from
                     customer left outer join orders on
                            c_custkey = o_custkey
                            and o_comment not like '%unusual%deposits%'
              group by
                     c_custkey
       )
group by
       c_count
order by
       custdist desc,
       c_count desc;
exit
```

Analysis: Query 13 is based on a sub-query the of second largest table (orders) and a moderate size table (customer), with the outer left join over the customer table, it forces to all customers to be considered along side of a counting of how many orders each one have; therefore all rows of both tables are fully processed. One third of the customers do not have any order. The final number of resulting rows is a small set between 0 and the maximum number of orders placed by customer.

Q1313. Customer Distribution Query (modified version of Q13)

```
-- using 472029144 as a seed to the RNG
set echo on
set feed on
alter session set NLS_DATE_FORMAT='YYYY-MM-DD';
create view q13 as
       (Select c_custkey as c_count,     count(o_orderkey) as how_many
              from
                     customer left outer join orders on
                            c_custkey = o_custkey
                            and o_comment not like '%unusual%deposits%'
              group by
                     c_custkey
       );
-- spool query1313.lst;
select
       c_count,
       count(*) as custdist
from q13
group by
       c_count
order by
       custdist desc,
       c_count desc;
drop view q13;
exit
```

Q14. Promotion Effect Query

This query monitors the market response to a promotion such as TV advertisements or a special campaign.

The Promotion Effect Query determines what percentage of the revenue in a given year and month was derived from promotional parts. The query considers only parts actually shipped in that month and gives the percentage. Revenue is defined as (l_extendedprice * (1-l_discount)).

```
-- @(#)14.sql 2.1.8.1
-- TPC-H/TPC-R Promotion Effect Query (Q14)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
set echo on
spool salida14.lis
-- query 14

select
	100.00 * sum(case
		when p_type like 'PROMO%'
			then l_extendedprice * (1 - l_discount)
		else 0
	end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
	lineitem,
	part
where
	l_partkey = p_partkey
	and l_shipdate >= date '1997-03-01'
	and l_shipdate < date '1997-03-01' + interval '1' month;
exit
```

Analysis: Query 14 calculates the numerator and the denominator of a fraction at the same time by using the CASE syntax to scan and process the data in one single sweep. A two-table join is required, with only 1/84th (one month) of the largest table qualifying for the join based on a date.

Q15. Top Supplier Query

This query determines the top supplier so it can be rewarded, given more business, or identified for special recognition.

The Top Supplier Query finds the supplier who contributed the most to the overall revenue for parts shipped during a given quarter of a given year. In case of a tie, the query lists all suppliers whose contribution was equal to the maximum, presented in supplier number order.

```
-- @(#)15.sql 2.1.8.1
-- TPC-H/TPC-R Top Supplier Query (Q15)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
set echo on
spool salida15.lis

-- query 15

create view revenue0 (supplier_no, total_revenue) as
       select
              l_suppkey,sum(l_extendedprice * (1 - l_discount))
       from
              lineitem
       where
              l_shipdate >= date '1997-02-01'
              and l_shipdate < date '1997-02-01' + interval '3' month
       group by
              l_suppkey;

select s_suppkey,   s_name, s_address, s_phone, total_revenue
from   supplier,revenue0
where
       s_suppkey = supplier_no
       and total_revenue = (
              select
                     max(total_revenue)
              from
                     revenue0
       )
order by
       s_suppkey;

drop view revenue0;

exit
```

Analysis: Query 15 requires either a view with aggregates or a temporary table. In the view, 1/28th of the Lineitems (3 months) are selected based on date and aggregated on Supplier. Close to all the Suppliers in the database will participate as output from this first step aggregation. The query itself returns details from the Supplier row which represents the maximum revenue. One row is returned from this query.

Q16. Parts/Supplier Relationship Query

This query finds out how many suppliers can supply parts with given attributes. It might be used, for example, to determine whether there are a sufficient number of suppliers for heavily ordered parts.

The Parts/Supplier Relationship Query counts the number of suppliers who can supply parts that satisfy a particular customer's requirements. The customer is interested in parts of eight different sizes as long as they are not of a given type, not of a given brand, and not from a supplier who has had complaints registered at the Better Business Bureau. Results must be presented in descending count and ascending brand, type, and size.

```
-- @(#)16.sql 2.1.8.1
-- TPC-H/TPC-R Parts/Supplier Relationship Query (Q16)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
-- query 16

select
	p_brand,p_type,p_size,
	count(distinct ps_suppkey) as supplier_cnt
from
	partsupp,
	part
where
	p_partkey = ps_partkey
	and p_brand <> 'Brand#22'
	and p_type not like 'MEDIUM PLATED%'
	and p_size in (21, 1, 27, 11, 33, 23, 32, 26)
	and ps_suppkey not in (
		select
			s_suppkey
		from
			supplier
		where
			s_comment like '%Customer%Complaints%'
	)
group by
	p_brand, p_type, p_size
order by
	supplier_cnt desc,
	p_brand, p_type, p_size;
exit;
```

Analysis: Query 16 is a correlated sub-query where the inner query excludes a very small number of suppliers (0.05%) which have customer complaints. The outer Query joins two medium size tables and returns one third of the brands and the different product sizes he has chosen alongside with the count of good suppliers for such parts.

Q17. Small-Quantity-Order Revenue Query

This query determines how much average yearly revenue would be lost if orders were no longer filled for small quantities of certain parts. This may reduce overhead expenses by concentrating sales on larger shipments.

The Small-Quantity-Order Revenue Query considers parts of a given brand and with a given container type and determines the average lineitem quantity of such parts ordered for all orders (past and pending) in the 7-year database. It considers what would be the average yearly gross (undiscounted) loss in revenue if orders for these parts with a quantity of less than 20% of this average were no longer taken.

```
-- @(#)17.sql 2.1.8.1
-- TPC-H/TPC-R Small-Quantity-Order Revenue Query (Q17)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
set echo on
spool salida17.lis
-- query 17

Select sum(l_extendedprice) / 7.0 as avg_yearly
from
       lineitem, part
where
       p_partkey = l_partkey
       and p_brand = 'Brand#23'
       and p_container = 'SM CAN'
       and l_quantity < (
              select
                     0.2 * avg(l_quantity)
              from
                     lineitem
              where
                     l_partkey = p_partkey
       );
Exit
```

Analysis: Query 17 uses correlated sub-query functionality to perform what-if analysis. A join of Lineitem and Part take place. 1/1000th of the rows in the Part table qualify based on the selection criteria. The item quantity that represents 20% of the average item quantity is calculated in the sub-query with the AVERAGE aggregation.

Q18. Large Volume Customer Query

The Large Volume Customer Query ranks customers based on their having placed a large quantity order. Large quantity orders are defined as those orders whose total quantity is above a certain level.

The Large Volume Customer Query finds a list of the top 100 customers who have ever placed large quantity orders. The query lists the customer name, customer key, the order key, date and total price and the quantity for the order.

Return the first 100 selected rows.

```
-- @(#)18.sql 2.1.8.1
-- TPC-H/TPC-R Large Volume Customer Query (Q18)
-- Function Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
-- query 18

select
       c_name,
       c_custkey,
       o_orderkey,
       o_orderdate,
       o_totalprice,
       sum(l_quantity)
from
       customer, orders, lineitem
where
       o_orderkey in (
              select
                     l_orderkey
              from
                     lineitem
              group by
                     l_orderkey
             having sum(l_quantity) > 313
       )
       and c_custkey = o_custkey
       and o_orderkey = l_orderkey

group by
       c_name,c_custkey,o_orderkey,o_orderdate,o_totalprice
order by
       o_totalprice desc, o_orderdate;
exit
```

Analysis: Query 18 this query returns 100 rows after joining 3 of the larger tables of the database. It consists of correlated sub-queries where the inner query fully scans the largest table to aggregate the quantities ordered. In order to produce the results of the outer query all the projected columns of the query need to be aggregated by the grouping function or by the sum operator.

Q19. Discounted Revenue Query

The Discounted Revenue Query reports the gross discounted revenue attributed to the sale of selected parts handled in a particular manner. This query is an example of a code which might be produced programmatically by a data mining tool.

The Discounted Revenue query finds the gross discounted revenue for all orders for three different types of parts that were shipped by air or delivered in person. Parts are selected based on the combination of specific brands, a list of containers, and a range of sizes.

```
-- @(#)19.sql 2.1.8.1
-- TPC-H/TPC-R Discounted Revenue Query (Q19)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
-- query 19
select
      sum(l_extendedprice* (1 - l_discount)) as revenue
from
      lineitem, part
where
      (
      p_partkey = l_partkey
      and p_brand = 'Brand#11'
      and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
      and l_quantity >= 9 and l_quantity <= 9 + 10
      and p_size between 1 and 5
      and l_shipmode in ('AIR', 'AIR REG')
      and l_shipinstruct = 'DELIVER IN PERSON'
      )
      or
      (
      p_partkey = l_partkey
      and p_brand = 'Brand#33'
      and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
      and l_quantity >= 17 and l_quantity <= 17 + 10
      and p_size between 1 and 10
      and l_shipmode in ('AIR', 'AIR REG')
      and l_shipinstruct = 'DELIVER IN PERSON'
      )
      or
      (
      p_partkey = l_partkey
      and p_brand = 'Brand#11'
      and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
      and l_quantity >= 20 and l_quantity <= 20 + 10
      and p_size between 1 and 15
      and l_shipmode in ('AIR', 'AIR REG')
      and l_shipinstruct = 'DELIVER IN PERSON'
      );
exit
```

Analysis: Query 19 joins the largest table of the database with a lookup table; the query conditions force to fully scan the part table and less than 1% of the LineItem table. The resulting set is then affected by an aggregation operator and returns just one row.

Q20. Potential Part Promotion Query

The Potential Part Promotion Query identifies suppliers in a particular nation having selected parts that may be candidates for a promotional offer.

The Potential Part Promotion query identifies suppliers who have an excess of a given part available; an excess is defined to be more than 50% of the parts, like the given part that the supplier shipped in a given year for a given nation. Only those parts whose names share a certain naming convention are considered.

```
-- @(#)20.sql 2.1.8.1
-- TPC-H/TPC-R Potential Part Promotion Query (Q20)
-- Function Query Definition. Approved February 1998
select
       s_name,s_address
from   supplier,nation
where  s_suppkey in (
              select
                     ps_suppkey
              from
                     partsupp
              where
                     ps_partkey in (
                            select
                                   p_partkey
                            from
                                   part
                            where
                                   p_name like 'chiffon%')
              and ps_availqty > (
                     select
                            0.5 * sum(l_quantity)
                     from
                            lineitem
                     where
                             l_partkey = ps_partkey
                         and l_suppkey = ps_suppkey
                         and l_shipdate >= date '1995-01-01'
                         and l_shipdate<date '1995-01-01'+interval '1' year)
       )
       and s_nationkey = n_nationkey
       and n_name = 'KENYA'
order by
       s_name;
exit
```

Analysis: Query 20 it returns two attributes after using 5 of the 8 tables of the database in four nested sub-queries. The inner most sub-query scans 1/7 of the largest database table (Lineitem) and applies and aggregation operator. The following sub-query uses 1% of the parts table to restrict more the result set of the inner-most sub-query; after that the result set is combined with the PartSupplier table, finally the outer most query works over two small tables (supplier and nation) to display the two selected attributes. It returns around 2% of the existing suppliers.

Q21. Suppliers Who Kept Orders Waiting Query

This query identifies certain suppliers who were not able to ship required parts in a timely manner. The Suppliers Who Kept Orders Waiting query identifies suppliers, for a given nation, whose product was part of a multi-supplier order (with current status of 'F') where they were the only supplier who failed to meet the committed delivery date. Return the first 100 selected rows.

```
-- @(#)21.sql 2.1.8.1
-- TPC-H/TPC-R Suppliers Who Kept Orders Waiting Query (Q21)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
-- query 21

select
       s_name,
       count(*) as numwait
from
       supplier, lineitem l1,
       orders, nation
where
       s_suppkey = l1.l_suppkey
       and o_orderkey = l1.l_orderkey
       and o_orderstatus = 'F'
       and l1.l_receiptdate > l1.l_commitdate
       and exists (
             select
                    *
             from
                    lineitem l2
             where
                    l2.l_orderkey = l1.l_orderkey
                    and l2.l_suppkey <> l1.l_suppkey
       )
       and not exists (
             select *
             from
                    lineitem l3
             where
                    l3.l_orderkey = l1.l_orderkey
                    and l3.l_suppkey <> l1.l_suppkey
                    and l3.l_receiptdate > l3.l_commitdate
       )
       and s_nationkey = n_nationkey
       and n_name = 'JORDAN'
group by
       s_name
order by
       numwait desc,
       s_name;
exit
```

Analysis: Query 21 correlates 3 sub-queries and returns 100 rows after using 6 tables, it is important to highlight that the largest database table (Lineitem) is used 3 times in the query resolution (L1, L2, L3 aliases).

Q22. Global Sales Opportunity Query

The Global Sales Opportunity Query identifies geographies where there are customers who may be likely to make a purchase.

This query counts how many customers within a specific range of country codes have not placed orders for 7 years but who have a greater than average "positive" account balance. It also reflects the magnitude of that balance. Country code is defined as the first two characters of c_phone.

```
-- @(#)22.sql 2.1.8.1
-- TPC-H/TPC-R Global Sales Opportunity Query (Q22)
-- Functional Query Definition
-- Approved February 1998
-- using 472029144 as a seed to the RNG
set echo on
spool salida22.lis

-- query 22

select
	cntrycode,
	count(*) as numcust,
	sum(c_acctbal) as totacctbal
from
	(
	select
		substring(c_phone,1,2)  cntrycode, c_acctbal
	from	customer
	where
		substring(c_phone,1,2) in('27','16','20','31','25','18','17')
		and c_acctbal > (
			select
				avg(c_acctbal)
			from
				customer
			where
				c_acctbal > 0.00
				and substring(c_phone,1,2) in
					('27','16','20','31','25','18','17')
		)
		and not exists (
			select
				*
			from
				orders
			where
				o_custkey = c_custkey
		)
	)  custsale
group by
	cntrycode
order by
	cntrycode;
exit
```

Analysis: Query 22 this is a 4 sub-queries correlation using 2 tables (orders and customers). The final results of the query are 7 rows which are the selected country codes. The inner most sub-query considers the whole orders table, but only 25% of the customers have orders, then the next sub-query restricts to the half of such customers who does not have orders by checking if the account balance is positive for those customers within the selected country codes, the following query restricts over the selected country codes and the outer most query aggregates over the 3 selected attributes.

Q2222. Global Sales Opportunity Query (modified version of Q22)

```
-- using 472029144 as a seed to the RNG
set echo off
set feed off
alter session set NLS_DATE_FORMAT='YYYY-MM-DD';
create table last_q (cntrycode char(2),
                         c_acctbal decimal(15,2));
insert into last_q (cntrycode,c_acctbal)
              select  substr(c_phone,1,2), c_acctbal
              from
                    customer
              where
                substr(c_phone,1,2) in ('27','16','20','31','25','18','17')
              and c_acctbal > (
                     select
                            avg(c_acctbal)
                     from
                            customer
                     where
                            c_acctbal > 0.00
                        and substr(c_phone,1,2) in
                                   ('27','16','20','31','25','18','17')
                        )
                        and not exists (
                                select
                                        *
                                from
                                        orders
                                where
                                        o_custkey = c_custkey
                        );
set echo on
set feed on
-- spool query2222.lst;
select cntrycode, count(*) as numcust, sum(c_acctbal) as totacctbal
from last_q
group by cntrycode
order by cntrycode;
drop table last_q;
exit
```

4.5 Scale Factor Definition

Within the TPC-H definition, the size of the Database (Data Warehouse) can be varied with an input parameter to the *dbgen* program. This size is called the “Scale Factor” and the number of rows for each table varies according to this scale factor. A Scale Factor of 1 –in approximate terms- generates a 1 GB Data Warehouse. Table 5 presents the number of rows generated during the current experiments with a Scale Factor (SF) equal to one. These data were used so that all the evaluated models have exactly the same data.

Table 5 Number of rows and table sizes for an SF=1

Table Name	Number of rows	Size (MB)
Customer	150,000	23.2
Lineitem	6,000,000	724.7
Nation	25	0.002
Orders	1,500,000	164.0
Part	200,000	23.1
PartSupplier	800,000	113.3
Region	5	0.0004
Supplier	10,000	1.3
	Total Size:	**1,049.60**

4.6 Extensions to the TPC-H Benchmark

TPC-H considers times to load data and execute queries, but it does not consider other tasks:

- Extraction times from the source systems
- Transformation times to conform with the target data model
- Input files sizes measurement
- Data Density
- Query execution times (in pristine mode)
- Data Warehouse Backup time
- Backup size
- Data Warehouse restore time

These tasks are important in Data Warehouse environments, because they are executed in Data Warehouse production environments and should be measured; for these reasons it has been considered important to extend the TPC-H [Gonzalez, 2006b].

The proposed extensions follow the philosophy of the TPC-H Power test [TPCH, 2002], in which all queries are executed sequentially. In Table 6, a comparison between the metrics considered in the extended TPC-H and the standard TPC-H benchmarks is presented. A database created without using any auxiliary performance structure will be referred to as the *pristine mode*.

Table 6 TPC-H and Extended TPC-H pristine models metrics

Metric	**Extended TPC-H**	**TPC-H**
Extraction times from transactional systems	Yes	No
Transformation times to conform to the target data model	Yes	No
Input files sizes measurement	Yes	No
Load data times into the Data Warehouse	Yes	Yes
Database table sizes after load (Database size)	Yes	Yes
Data Density Measurement	Yes	No
Queries execution times (Pristine mode)	Yes	No
Data Warehouse Backup time	Yes	No
Backup size	Yes	No
Data Warehouse restore time	Yes	No

There is another set of metrics which is useless while evaluating data models, because they are technology improvements over the relational technology (indexing, statistics computation and query optimizer effects). However, these were measured to provide pragmatic performance metrics for real Data Warehouse environments (Table 7).

The proposed extensions have been made in order to measure Data Density (objective 1) and in order to be able to understand the total impact of the evaluated alternative data models. Those are the metrics to be evaluated during the experiments, and they will assist in choosing the best alternative data model for Data Warehouse environments. In

Chapter 6, the results measurements of these metrics are presented according to the different test scenarios that were executed.

Table 7 TPC-H and Extended TPC-H technology specific metrics

Metric	Extended TPC-H	TPC-H
Index creation times	Yes	Yes
Index sizes	Yes	Yes
Query times (with indexes)	Yes	No
Statistics computation times	Yes	Yes
Query times (with indexes & statistics)	Yes	Yes
Query times (with statistics without indexes)	Yes	No

4.7 Conclusions

There exist different benchmarks to be used in Data Warehouse environments. None of them covers all the aspects of a Data Warehouse. For this reason, the TPC-H has been chosen as it has wide acceptance in the research and industry communities. However as it does not cover all the aspects required on a Data Warehouse environment, it has been extended with a series of metrics which will cover the full Data Warehouse cycle, and will be used to execute and measure the experiments of this research.

From the extensions made to the TPC-H, the metrics have been classified in order to identify whether they are useful to evaluate the underlying data model of the DBMS or if they are useful for evaluating technology aspects.

The extensions to the TPC-H benchmark are a contribution made to the research community during this thesis, and have been reported in Gonzalez [2006b].

CHAPTER 5
Implementation of the Storage Mechanism for the Transrelational^{TM} Model

In this chapter, an implementation of the storage mechanism for the Transrelational^{TM} model based on the public domain definition provided by C.J. Date in Date [2004] is presented. Such implementation has been reported in Gonzalez [2006a] as the first public domain instantiation of the storage mechanism for the model. After presenting the implementation, the results of performance tests utilising a set of metrics for Data Warehouse environments are presented, and are compared against a traditional n-ary Relational implementation.

As already indicated, the only public domain description of the Transrelational^{TM} model is provided by Date. First, some of the basic definitions of the model are introduced in order to establish a baseline for the experimental consideration; all quotations and references to Date are attributable to Date [2004].

As a starting point, Date states that *"The Transrelational^{TM} (TR) is not intended as a replacement for the Relational model"*, from which it can be seen as an alternative route to implementing the relational model and thus to building a Relational DBMS. In this case, "trans" stands for transform; it is not intended to mean "beyond".

From the early days of data processing systems, through the development of relational databases and up to the present day, data have been conceived predominantly as Records of *n* fields or attributes. This approach has been called an n-*ary Storage Model* (NSM) [Copeland, 1985] or, in Date's nomenclature, *Direct Image Systems* (DIS). Within this approach, data are seen, stored and processed following a horizontal approach (rows).

Alternatively, there is also a vertical approach (columns) to store and process data, and this has its origins in Copeland's seminal paper *"A Decomposition Storage Model"* (DSM) [Copeland, 1985].

Transrelational differs from both vertical and horizontal approaches, but as in the *Field Values Table* (FVT), each column stores distinct values, and most of the processing can be done at column level; therefore, Transrelational is closer to a vertical approach.

5.1 Introduction to the Transrelational Model

The Transrelational^{TM} model was defined by Steve Tarin, and patented in the United States of America [US Patent, 1999]; and it has been promoted to the Database community by C.J. Date through a series of seminars and in the latest edition of his widely-adopted textbook [Date, 2004]. However, as far as has been determined, there is no implementation available for either commercial or research use. Therefore, in order to carry out the experimental consideration of this research, the general description made by Date [2004] of the Transrelational^{TM} model and its behaviour has been utilised to implement the essential algorithms that make up the storage mechanism for the model. Since Date [2004] has provided the only public domain documentation of the model, henceforward referred to as TR following his nomenclature, Date’s work will be referenced extensively while describing the implementation and executing the experiment.

According to Date [2004], the TR model has an implicit data compression mechanism by using *condensed columns;* this eliminates the duplicate values at a column level. This is very appealing in Data Warehouse environments where it is common to have many repetitive values at column level; this will reduce sparsity and increase data density.

5.2 Model Data Structures

The model structures were described briefly in section 3.5.2, but for clarity a description of them is made here again. To illustrate the characteristics of the model, the examples developed by Date [2004] have been utilised. The Transrelational model consists basically of two storage structures: the *Field Values Table (FVT)* Figure 30 (b), in which each column contains the values from the corresponding field of the input file rearranged in an ascending sort order, and the *Record Reconstruction Table (RRT)* Figure 30(C), which keeps a set of values derived from the original input file that can be thought of as pointers to enable the original record to be rebuilt when necessary using the *Zig Zag* algorithm [Date, 2004] and which has been literally reproduced in section 5.3 from Date [2004].

Record Sequence	S#	SNAME	STATUS	CITY
1	S4	Clark	20	London
2	S5	Adams	30	Athens
3	S2	Jones	10	Paris
4	S1	Smith	20	London
5	S3	Blake	30	Paris

	S#	SNAME	STATUS	CITY
1	S1	Adams	10	Athens
2	S2	Blake	20	London
3	S3	Clark	20	London
4	S4	Jones	30	Paris
5	S5	Smith	30	Paris

	S#	SNAME	STATUS	CITY
1	5	4	4	5
2	4	5	2	4
3	2	2	3	1
4	3	1	1	2
5	1	3	5	3

Figure 30 (a) A Suppliers Relation (b) Field Values Table (c) Record Reconst. Tab.

5.3 The Zig Zag Algorithm

To understand how both tables are used when rebuilding a record utilising the Zig Zag algorithm according to the description made in Date [2004], mentioned in section 3.5.2, for purposes of clarity it has been reproduced literally here:

"Step l: Go to cell [1, 1] of the Field Values Table and fetch the value stored there: namely, the supplier number S1. That value is the first field value (that is the S# field value) within a certain supplier record in the suppliers file.
Step 2: Go to the same cell (that is, cell [1, 1]) of the Record Reconstruction Table and fetch the value stored there: namely, the row number 5. That row number is interpreted to mean that the next field value (which is to say, the second or SNAME value) within the supplier record whose S# field value is S1 is to be found in the SNAME position of the fifth row of the Field Values Table -in other words, in cell (5,2) of the Field Values Table. Go to that cell and fetch the value stored there (supplier name Smith).
Step 3: Go to the corresponding Record Reconstruction Table cell [5, 2] and fetch the row number stored there (3). The next (third or STATUS) field value within the supplier record we are reconstructing is in the STATUS position in the third row of the Field Values Table-in other words, in cell [3, 3]. Go to that cell and fetch the value stored there (status 20).
Step 4: Go to the corresponding Record Reconstruction Table cell [3, 3] and fetch the value stored there (which is 3 again). The next (fourth or CITY) field value within the supplier record we are reconstructing is in the CITY position in the third row of the

Field Values Table - in other words, in cell [3, 4]. Go to that cell and fetch the value stored there (city name London).

Step 5: Go to the corresponding Record Reconstruction Table cell [3, 4] and fetch the value stored there (1). Now, the "next" field value within the supplier record we are reconstructing looks like it ought to be the fifth such value; however, supplier records have only four fields, so that "fifth" wraps around to become the first. Thus, the "next" (first or S#) field value within the supplier record we are reconstructing is in the S# position in the first row of the Field Values Table-in other words, in cell [1, 1]. But that is where we came in, and the process stops."

To this point, the model provides no potential database size reduction because all values are kept, and additional data are held in the Record Reconstruction Table; this version will be referred to as TR Version 1.

The desired reduction is achieved when the *Condensed Columns* are introduced. As can be observed in Figure 30(b), a considerable amount of redundant data is stored; this is also true in traditional n-ary systems (see Figure 30(a)). The Column-Condensing process aims to eliminate such redundancy by keeping unique values at column level; it will be referred to as TR Version 2.1.

This process should be applied selectively, since attempting to condense columns already consisting of unique values does not make sense. However, as each value can be used in many "records", it is necessary to keep Row Ranges (numbers in squared brackets in Figure 31) to avoid losing information on how to reconstruct the record; this will be referred to as TR Version 2.3. The resulting FVT with condensed columns and row ranges is presented in Figure 31**.**

	S#	SNAME	STATUS	CITY
1	S1	Adams	10 [1:1]	Athens [1:1]
2	S2	Blake	20 [2:3]	London [2:3]
3	S3	Clark	30 [4:5]	Paris [4:5]
4	S4	Jones		
5	S5	Smith		

Figure 31 Condensed Field Values Table with row ranges

The Column-Condensing process destroys the unary relationship between the cells of the FVT and the cells in the Record Reconstruction Table (RRT), but the Zig Zag algorithm is easily adaptable as stated in Date [2004].

"Consider cell [i, j] of the Record Reconstruction Table. Instead of going to cell [i, j] of the Field Values Table, go to cell [i', j] of that table where cell [i', j] is that unique cell within column j of that table that contains a row range that includes row i."

5.4 Implementation

During the experiments carried out in this thesis, algorithms to create the Field Values Table, the Record Reconstruction Table and the Zig Zag algorithm to rebuild the records were implemented. Some variations and improvements were made during implementation and these will be described and discussed in the following subsections.

This implementation was focused on the initial bulk load of the Data Warehouse, and it retained the limitations identified by Date [2004] in which updates are discarded and the Database is Read Only. Inherently, Data Warehouse environments are more suited to these assumptions (with batch updates during off line hours and read only during operation hours) than transactional systems. Consequently, it could be argued that the benchmarking of the TR model for Data Warehouse environments is both relevant and important.

An extraction tool was written (see section 6.4) in order to generate the flat files that will be processed by the TR model algorithms. One important point that was introduced during the extraction process is that each record of the flat file is pre-appended with its record number to provide additional support during the Transformation and Loading stages.

5.4.1 The Field Values Table

Data within each column are rearranged into ascending sort order. All operations are made in bulk and parallelised as much as possible; extensive use of the sort, parallelisation and synchronisation mechanisms offered by the operating system has been made [Gilly, 1994]. The sorting process of each column is made in parallel as each column is independent. Improvements were introduced in order to prepare the creation of the RRT and minimise reprocessing. The first step is to create a structure that will be called the *Working File* (WKF) which is an enhancement to the algorithms described by Date, on which each column is in ascending order and maintains the original record number as sub-index Figure 33(a). The portion of the program that computes the WKF is shown in Figure 32.

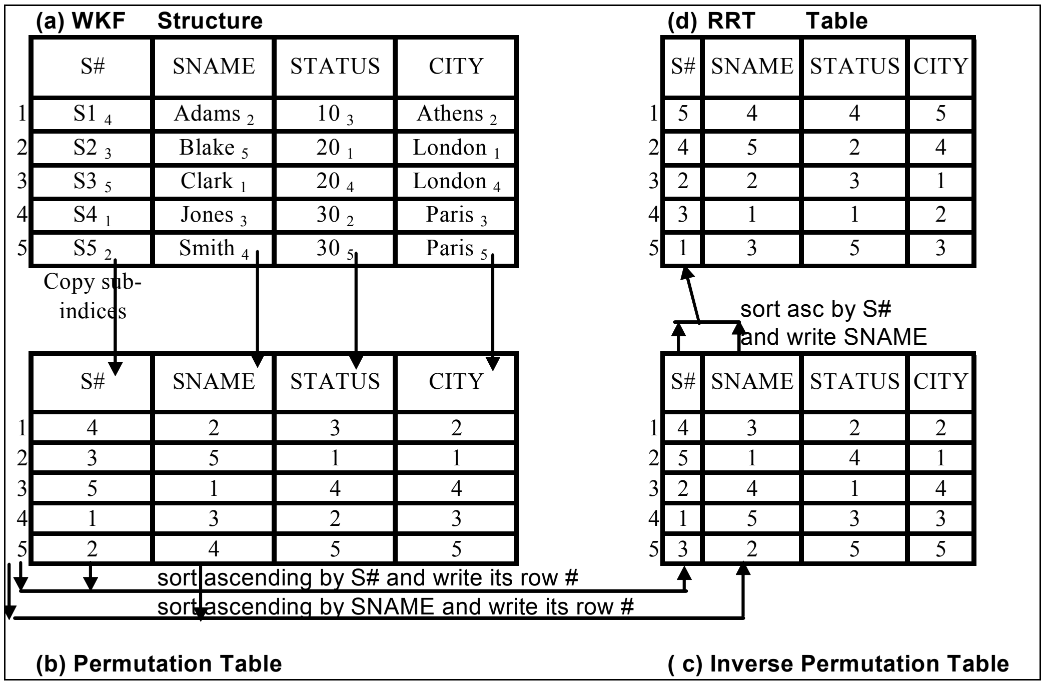
```
###########################################################################
### Computing the FVT table. The temporally results are in files .wkf    ###
###########################################################################
CONCAT_STRING=' '
          for ((i =2 ; i<=$NUM_COLS; i++ ))
            do
            ## separate each column with its row number (col_1 + col_i), ##
            ## then sort each column in numeric order by the i-column    ##
            ## Each column is processed in background in order to       ##
            ## parallelize                                               ##

              cut -f$i $LOADING_DATA_FILES/$1| paste
$TRANSREL_TMP/$1"_col1.wkf" - | sort +1 > $TRANSREL_TMP/$1"_col$i.wkf" &
              CONCAT_STRING=$CONCAT_STRING' '$TRANSREL_TMP/$1"_col$i.wkf"
#keeping file names to paste later
            done
            wait # to synchronize and wait until all files (columns) are
done #
            echo $CONCAT_STRING
            paste $TRANSREL_TMP/$1"_col1.wkf" $CONCAT_STRING >
$TRANSREL_TMP/$1_tmp1.wkf
            rm $CONCAT_STRING
```

Figure 32 FVT Computation Program

From the WKF structure, the Field Values Table with condensed columns (Figure 31) is derived by choosing unique values in ascending order and recording the corresponding row ranges. The other structure derived from the WKF structure is the Permutation Table [Date, 2004] Figure 33(b), which is required to build the Record Reconstruction Table.

The Column-Condensing process is selective, as recommended by Date [2004]. In the current implementation, it has been established that columns with unique values and those where the reduction would be lower than the established condensation threshold are not condensed.

The condensation threshold is not part of Date's algorithms, and has been defined as the ratio between the original number of values (including repetitions) and the number of unique values. The condensation threshold is 30%; and this was set after testing different threshold values and considering the balance between the disk space required to keep row ranges versus the disk space required to keep the complete column with relatively few repetitive values. The processing time was also considered to set the threshold. This approach was taken in order to maintain a balance between the theoretical model and the pragmatic implementation.


(a) WKF Structure
(d) RRT Table
Copy sub-indices
sort asc by S# and write SNAME
sort ascending by S# and write its row #
sort ascending by SNAME and write its row #
(b) Permutation Table
(c) Inverse Permutation Table

(a) WKF Structure

	S#	SNAME	STATUS	CITY
1	S1 $_{4}$	Adams $_{2}$	10 $_{3}$	Athens $_{2}$
2	S2 $_{3}$	Blake $_{5}$	20 $_{1}$	London $_{1}$
3	S3 $_{5}$	Clark $_{1}$	20 $_{4}$	London $_{4}$
4	S4 $_{1}$	Jones $_{3}$	30 $_{2}$	Paris $_{3}$
5	S5 $_{2}$	Smith $_{4}$	30 $_{5}$	Paris $_{5}$

(d) RRT Table

	S#	SNAME	STATUS	CITY
1	5	4	4	5
2	4	5	2	4
3	2	2	3	1
4	3	1	1	2
5	1	3	5	3

(b) Permutation Table

	S#	SNAME	STATUS	CITY
1	4	2	3	2
2	3	5	1	1
3	5	1	4	4
4	1	3	2	3
5	2	4	5	5

(c) Inverse Permutation Table

	S#	SNAME	STATUS	CITY
1	4	3	2	2
2	5	1	4	1
3	2	4	1	4
4	1	5	3	3
5	3	2	5	5

Figure 33 Proposed Alternative Algorithm to Build the Record Reconstruction Table

5.4.2 The Record Reconstruction Table

A permutation is defined by Date [2004] as the ordering of the records by a particular column. For example, the permutation corresponding to ascending S# ordering is 4,3,5,1,2 (refer to the sub-indices on Figure 33(a)). According to this definition, the *Permutation Table* is computed directly from the original input file, but instead of doing it in this way it has been derived it from the WKF structure by using the already ascending ordered values but taking the sub-indices (see Figure 33(a) and (b)). Thus the columns of the Permutation Table can be computed in parallel. The first column of the Permutation Table keeps an ascending sequence from 1 to the number of records in the table (i.e. the row number within the Permutation Table). These are some of the improvements made during this research to Date's algorithm.

Date also defined the *Inverse Permutation* thus: *"That unique permutation that if applied to the original sequence 4,3,5,1,2, will produce the sequence 1,2,3,4,5"*. It is computed column by column on the permutation table by applying the previous rule to obtain the ascending sequence 1,2,3,4,5.

During this research, it was found that it is more efficient to compute the *Inverse Permutation Table* from the existing Permutation Table by taking each column together with its corresponding row number, and then sorting ascending by the corresponding column values and writing the row number in the Inverse Permutation Table rather than the column values (in this example S#) (see Figure 33(b) and (c)). The resulting Inverse Permutation Table is exactly the same as that described by Date, and its columns can be computed in parallel. This is another improvement to Date's algorithm.

Finally, the Record Reconstruction Table can be built from the Inverse Permutation Table. Date's algorithm [Date, 2004] is as follows:

"Go to the cell [i, 1] of the Inverse Permutation Table. Let that cell contain the value r; also the next cell to the right, cell [i,2], contain the value r'. Go to the r-th row of the Record Reconstruction Table and place the value r' in cell [r,1]." Following the algorithm for i=1,2,3,4,5 will produce the entire S# column of the Record Reconstruction Table. The other columns are computed in a similar way.

This algorithm processes one row at a time, which is inefficient; therefore the following algorithm to compute the Record Reconstruction Table is proposed:

From the Inverse Permutation Table use column i and column i+1, sort in ascending order by i-th column values and write the value held in i+1 column to the corresponding i-column in the RRT. This applies to all columns except the last one which must use the n-th column and the 1st column (see Figure 33 (c) and (d)).

The algorithm presented in this thesis enables bulk processing, and each column is processed in parallel as there is no dependency between columns and it is another improvement to Date's algorithm.

5.4.3 Implemented Versions

Four versions of the storage mechanism for the TR model were implemented in order to gather information regarding its behaviour with different characteristics. Each version, its characteristics, and the implications for the Zig Zag algorithm [Date, 2004] are listed in Table 8**.**

Table 8 Implemented Versions of the Storage Mechanism for the TR Model

Version 1	Field Values Table (FVT) keeps repetitive values. Both the Record Reconstruction Table (RRT) and the Zig Zag Algorithm are as stated in Date [2004].
Version 2.1	All columns in all tables are condensed in their corresponding FVT tables and the remaining unique values keep row ranges. The Zig Zag algorithm is enhanced to be aware of row ranges. The RRT Table remains unchanged.
Version 2.2	Only the Fact Table is considered to condense its columns. The Zig Zag algorithm needs to be improved to detect the Fact Table and be aware that row ranges only exist in the Fact Table but not in any other table. RRT remains unchanged.
Version 2.3	Selective column condensation in the FVT is applied if the established threshold is reached; this is applied to all tables as proposed by Date. The Zig Zag algorithm needs to be aware of condensed and uncondensed columns, and those which are condensed have row ranges. RRT Table remains unchanged.

5.5 Results and analysis of the Implemented Versions of the Storage Mechanism for the TR

The results presented in this section follow the flow of the Extraction Transformation and Loading Process. As mentioned before, a tool was made to extract data and generate the input flat files to be loaded in both Relational and TR instantiations. The differences between input flat file sizes are small but the TR input files are slightly bigger because of the extra column required to keep the row number that will be used for further processing. The resulting flat file sizes are presented in Table 9**.**

Table 9 Extracted file sizes to be used as input

Scale Factor (SF)	Relational (MB)	TR (MB)
100 MB	102.81	103.77
1GB	1,049.60	1067.46

The next step is to Transform and Load the input files into the instantiations for both models (n-ary-Relational and Transrelational). As stated before, four different versions of TR's storage mechanism were implemented (Table 8). The results for these versions are presented in Table 10 (SF=100MB) and Table 11 (SF=1GB).

Table 10 DBMS Object Size in MB with SF=100MB

TPC-H Table Name	Relational	**TR V1** (with repetitive values)	**TR V2.1** (everything condensed)	**TR V 2.2** (only Fact Table condensed)	**TR V2.3** (selective condensation)
Region	0.0625	0.0005	0.0005	0.0005	0.0005
Nation	0.0625	0.0024	0.0027	0.0024	0.0023
Supplier	0.1875	0.1600	0.1900	0.1600	0.1600
Customer	3.00	3.04	3.41	3.04	2.79
Part	3.00	3.41	2.34	3.41	2.12
PartSupplier	13.00	14.19	13.90	14.19	13.05
Orders	19.00	25.63	21.91	25.63	18.56
Lineitem	88.00	137.07	86.09	86.09	86.90
TOTAL	**126.31**	**183.50**	**127.84**	**132.52**	**123.58**

Table 11 DBMS Object Size in MB with SF=1GB

TPC-H Table Name	Relational	**TR V1** (with repetitive values)	**TR V2.1** (everything condensed)	**TR V 2.2** (only Fact Table condensed)	**TR V2.3** (selective condensatio n)
Region	0.0625	0.0005	0.0005	0.0005	0.0005
Nation	0.0625	0.0024	0.0027	0.0024	0.0023
Supplier	2.0	1.75	2.07	1.75	1.68
Customer	27.0	32.00	36.23	32.00	29.3
Part	30.0	36.37	24.74	36.37	21.52
PartSupplier	128.0	149.00	137.69	149.00	130.93
Orders	192.0	274.96	235.32	274.96	200.89
Lineitem	848.0	1489.71	925.73	925.73	962.22
TOTAL	**1,227.1**	**1,983.78**	**1,361.81**	**1,419.80**	**1,346.54**

TR Version 1 is of limited value because it keeps all repetitive values and adds more information within the RRT table; as a result it increases the n-ary-Relational instantiation DB size by a factor of around 50%.

Considering Table 10 and Table 11 with version 2.1 (in which all columns are condensed), and for medium-sized tables (Supplier and Customer), the effect of condensing resulted in bigger table sizes than the corresponding sizes without being condensed; this is as a result of keeping the row ranges; for the bigger tables (Lineitem and Orders) the Column-Condensing process is beneficial. Considering that the Lineitem Table (Fact Table) is the biggest table, Version 2.2 was introduced. The benefit of this version is that only the Fact Table is passed through the Column-Condensing process in order to reduce the CPU processing time, but the downside is that the final DB size is bigger than Version 2.1.

Finally, in version 2.3 all tables are processed by the Column-Condensing process but in a selective fashion in which if the estimated condensing level reaches the established threshold then the column is condensed, otherwise there is no benefit in investing CPU processing time which will not achieve significant column size reductions. According to the results obtained, version 2.3 is the one that offers a better balance between processing time and disk space consumption, but it requires a complex Zig Zag algorithm to rebuild records when necessary. The Zig Zag algorithm needs to be sufficiently intelligent to identify condensed columns and uncondensed columns and make use of row ranges when rebuilding the original record.

These analyses have been focused on the FVT since, as identified by Date [2004], the FVT will offer compression benefits. However, the experimental results show that, even when condensing repetitive values, the final database size is bigger or, at best, only slightly smaller than the traditional n-ary Relational instantiation.

Further analyses based on Version 2.3 show that the FVT behaves in keeping with Date's description, but the problem is with the RRT. While the RRT at first sight holds only integers, after millions of rows (i.e. Lineitem=6 million rows for SF=1GB), these integers are of more considerable size and occupy most of the final database space. These results are presented in Table 12 (SF=100MB) and Table 13 (SF=1GB).

Table 12 TR V2.3 RRTs and FVTs with SF=100MB.

TPC-H Table Name	Relational (MB)	TR V2.3 (MB)	RRT (MB)	FVT (MB)	% RRT vs. total	%FVT vs. total
Region	0.0625	0.0005	0.0001	0.0004	11%	89%
Nation	0.0625	0.0023	0.0003	0.0020	13%	87%
Supplier	0.1875	0.1600	0.0300	0.1300	19%	81%
Customer	3.00	2.79	0.68	2.11	24%	76%
Part	3.00	2.12	1.04	1.08	49%	51%
PartSupplier	13.00	13.05	2.68	10.37	21%	79%
Orders	19.00	18.56	8.95	9.61	48%	52%
Lineitem	88.00	86.90	66.36	20.54	76%	24%
TOTAL	126.31	123.58	**79.74**	**43.84**	**65%**	**35%**

Table 13 TR V2.3 RRTs and FVTs with SF=1GB

TPC-H Table Name	Relational (MB)	TR V2.3 (MB)	RRT (MB)	FVT (MB)	% RRT vs. total	%FVT vs. total
Region	0.0625	0.0005	0.00005	0.00045	10%	90%
Nation	0.06	0.0023	0.00030	0.00200	13%	87%
Supplier	2.0	1.68	0.37	1.31	22%	78%
Customer	27.0	29.3	8.06	21.24	28%	72%
Part	30.0	21.52	12.29	9.23	57%	43%
PartSupplier	128.0	130.93	31.41	99.52	24%	76%
Orders	192.0	200.89	51.69	149.20	26%	74%
Lineitem	848.0	962.22	760.34	201.88	79%	21%
TOTAL	1,227.1	1,346.54	**864.16**	**482.38**	**64%**	**36%**

As in any Data Warehouse environment, the Fact table (in this case LineItem) occupies most of the space; with SF=1GB this table occupies 962MB of 1,346MB of the total DB size. It is important to note, however, that the corresponding RRT for Lineitem required 760MB of those 962MB. In general, RRT structures are 65% of the total DB space, while the FVT structures are the remaining 35%. From these results, it is necessary to stress that further work is necessary to tackle the RRT structures, and particularly the RRT for Fact Tables, to enable the TR model to achieve the benefits predicted.

Another key finding of the experiment is that with the bigger scale factor, the Version 2.3 (selective condensation) has better results than any other version (see Table), including Version 2.1 in which every column is condensed, but remains very close (additional 10%) to the traditional n-ary Relational implementation.

Finally, another aspect to be considered is the time to Transform and Load the input files into the DBMS. In this aspect, the TR instantiation required more time than the n-ary-Relational instantiation. The Transformation and Loading time was not linear with respect to DB size, as presented in Table 14, with around 4 times more with SF=100MB and 10 times more with SF=1GB.

Table 14 The Transformation and Loading Times.

	Scale Factor (SF)	Transformation & Loading time
Relational	100MB	2.4 minutes
Transrelational^{TM}	100MB	10.2 minutes
Relational	1GB	19.9 minutes
Transrelational^{TM}	1GB	191.6 minutes

5.6 Transrelational Query Process

Until now Transrelational has been described from the load and storage characteristics, but it is necessary to describe its characteristics for querying data. The Query process is not fully described in [Date, 2004] therefore a general description of the querying process will be described here, using the general behaviour of Transrelational as described in [Date, 2004], which still thinking on managing data as records during the record reconstruction process and because it still considering the processing of all table's columns. A novel way to access data in a more efficient manner has been envisage in this thesis research. The novelty of the proposed querying method is a mix between the way Transrelational process data [Date, 2004] and the way the Binary-Relational model access data.

In Date [2004] a brief description of how Transrelational manages data its way to rebuild the records using the *fvt* and the *rrt* tables is made, but it always retrieve all columns of the table into memory and rebuilds the corresponding records. In contrast the novel way to process data should begin by parsing the Query statement (in this thesis the SQL language structure has been assumed as the query language) to identify the relations involved in the "*from"* clause, the columns that will be retrieved by the "*select"* clause, and the conditions that retrieved data should satisfy in the "*where*" clause. After that records are going to be reconstructed using the Zig Zag algorithm. A disadvantage has been found, as the method described in [Date, 2004] needs to process all columns and the complete fvt and rrt should be accessed during the query resolution. This will perform worst than the Binary-Relational model which only will access slices of information to solve the query by accessing only the columns involved in the *select* or *where* clauses of the query [Boncz, 2002].

If the approach suggested in [Date, 2004] is used, during the query optimisation process intermediate result sets need to be reconstructed until the final result set is obtained, therefore the advantage of storing data without duplicates on the fvt table is vanished after the first intermediate result set is generated. The benefit of the having data without duplicates at column level is utilised only in the first part of the query, but after that once records have been rebuild the general behaviour of the query resolution process is similar to the way actual RDBMS work by generating intermediate results sets until getting the final result set to be display.

In this research a novel way to solve queries has been envisage, the novelty of the query resolution process involves accessing only the columns involved in the query statement and avoid rebuild records in intermediate steps as far as it is possible. In the following paragraphs the way both methods will work is illustrated with an example.

The following query example will be used to explain both approaches to solve the query.

```
Select pnumber, pname
From parts
Where city = "London"
```

Figure 34 Sample Query

The fvt and the rrt are as follows:

pnumber	pname	colour	city
p1	Bolt [1:1]	Blue [1:2]	London [1:3]
p2	Cam [2:1]	Green [3:1]	Oslo [4:1]
p3	Cog [3:1]	Red [4:3]	Paris [5:2]
p4	Nut [4:1]		
p5	Screw [5:2]		
p6			

4	3	4	1
1	2	6	4
5	6	5	6
6	4	1	3
2	1	2	2
3	5	3	5

Figure 35 fvt and rrt For the Sample Query

To implement this query, it is necessary to do a binary search on column CITY of the fvt (Figure 35), looking for a cell containing the value "London"; note that such a cell must be unique if it exists at all, because the column is condensed. If the search fails, then the result of the query is empty and the process ends. In this case, it founds cell [1,4] with a row range [1,3] meaning that there are 3 rows that satisfy the condition of being in "London". Using the row range then 3 cells of the rrt will be involved on the records reconstruction, those will be [1,4], [2,4], [3,4]; each one of these 3 rrt's cells contain the row numbers for the "next" cell in the rrt table. Zig Zags can be reconstructed by following the appropriate pointer rings in the rrt table.

During the reconstruction of the record using Zig Zags, Date [2004] assumes that the fvt is a single file; therefore all columns of the fvt need to be processed. By contrast as result of this thesis research it has been envisage that it is not necessary to think the fvt as a single file, what it is proposed is to represent the fvt as a set of independent objects (columns). Having "n" files for a table of "n" columns. Similar to the way Binary-Relational manages independent relations of 2nd degree, but the fvt does not need to keep the key as the record linkage structure is keep on the rrt. By doing this column independent management, then when executing the Zig Zags, only the columns to be displayed are accessed during the record reconstruction process, saving disk and memory access.

Using the same select statement of Figure 34, the resolution is similar as described before but the Zig Zag algorithm should be modified to include the following condition in order to avoid retrieve and access all the columns.

```
If current_column_name is in the columns to be displayed in the select statement
Then retrieve the corresponding value
Else do not retrieve the value and do not open the corresponding column file. Instead
the Zig Zag must go to the next position on the rrt and continue the algorithm.
```

By doing this improvement to the Zig Zag algorithm, savings on data access and processing are achieved. If Transrelational or more precisely the Zig Zag is used without this modification, the performance of the TR would be lower than the performance of the Binary-Relational, because more disk blocks are read and more memory is used.

5.7 TR Conclusions

The Transrelational model is recent and as far as it could be determined there was no implementation available of this model; for this reason it has been necessary to implement the essential algorithms from the public domain definition made by C.J. Date. As a contribution of this thesis, the first public domain implementation of the storage mechanism for the Transrelational model has been made.

From the implementation and measurements made with the different versions of the storage mechanism for the TR Model, the version that offers the major benefits in terms of disk space usage and CPU processing time is Version 2.3 with a selective condensation, but even this version is not achieving the main benefits of objectives 1 and 2 of this research, which are related with achieving a higher Data Density. TR Version 2.3 will be used as the Transrelational instantiation in Chapters 6 and 7, in which all the alternative data models are compared.

CHAPTER 6
Experiments and Results

6.1 Introduction

This chapter describes the benchmarking environment (section 6.2), the experimental methodology (section 6.3.2), the metrics considered (section 6.3.3), and the results arising from each data model instantiation (section 6.4). Further analysis and deep discussion of such results are presented in Chapter 7

In order to benchmark the selected models and its instantiations, the extended TPC-H benchmark has been chosen (section 4.6). The tests were executed with two database sizes, called scale factors (SF=100 MB and SF=1GB) according to the TPC-H definition in section 4.4.

6.2 The Benchmarking Environment

The benchmarking environment includes the Hardware and Software components required to run the experiments; those components are described in sections 6.2.1 and 6.2.2.

The same hardware was used for each data model instantiation with the premise of having the same conditions and no hardware variations. Each data model instantiation was run alone during the benchmark test; the other models DBMS were shut down in order to avoid interference with the benchmarked model.

6.2.1 Hardware

The machine used to evaluate the defined metrics has one CPU Pentium IV with clock speed of 1.60GHz. 512 MB in RAM, Cache size 256 KB, Bus speed 100MHz.

6.2.2 Software

The Operating System installed in the computer was Linux Fedora 2 version 2.6.8-1.521.

The relational instantiation used is Oracle Version 9.0 with its corresponding SQL*Plus and SQL*Loader utilities.

The binary-relational instantiation used is MonetDB Version 4.4.0; the SQL utility version 2.4 and its loader module is called Ascii_io, which is invoked by the Load.mil utility.

The Associative / Triple Store instantiation used is SentencesDB version 3.5, and its loader utility is CSVImport version 3.5.

The Transrelational (TR) instantiation used is the one implemented during this research and which has been described in Chapter 5. The version used is TR version 2.3. SHQL version 1.3 [SHQL, 2005] is used to provide a SQL interface for the Transrelational model's DDL statements.

6.3 Experimental Methodology

The experimental methodology consists of extracting data from a horizontal repository and transforming it according to the format requirements of each alternative data model. Once data are transformed they are loaded into each data model instantiation and data density is measured in order to satisfy the research objectives 1, 2 and 3 (find an alternative data model which reduces data sparsity and increases data density).

Those data models with lower data density will be discarded, and only those with better data density will continue with further tests.

After loading, the data will be queried to measure the response time of each data model instantiation.

After the query execution, the data will be backed up and measurements will be made for this maintenance phase.

The best-suited model will be selected and it will be considered for use within a Data Warehouse architectural configuration (according to the research objective 4) which can perform properly covering the whole Data Warehouse cycle (section 6.3.1).

The experiments have been designed to follow the full Data Warehouse cycle approach, which groups the operations involved in any production Data Warehouse; this cycle is explained in the following section.

6.3.1 The Full Data Warehouse Cycle Definition

The full Data Warehouse (DWH) cycle consists of a series of phases involved in the operation of the Data Warehouse, beginning with extraction of data from transactional systems to its utilisation with query and OLAP tools.

The full Data Warehouse cycle is defined here as follows:

1. Data Extraction from the transactional systems
2. Data Transformation according to the targeted Data Warehouse
3. Data Loading into the Data Warehouse
4. Data Storing into the Data Warehouse
5. Data Optimisation (if required by the DBMS)
6. Data Querying from the Data Warehouse
7. Data Warehouse Maintenance

These phases are presented schematically in Figure 36

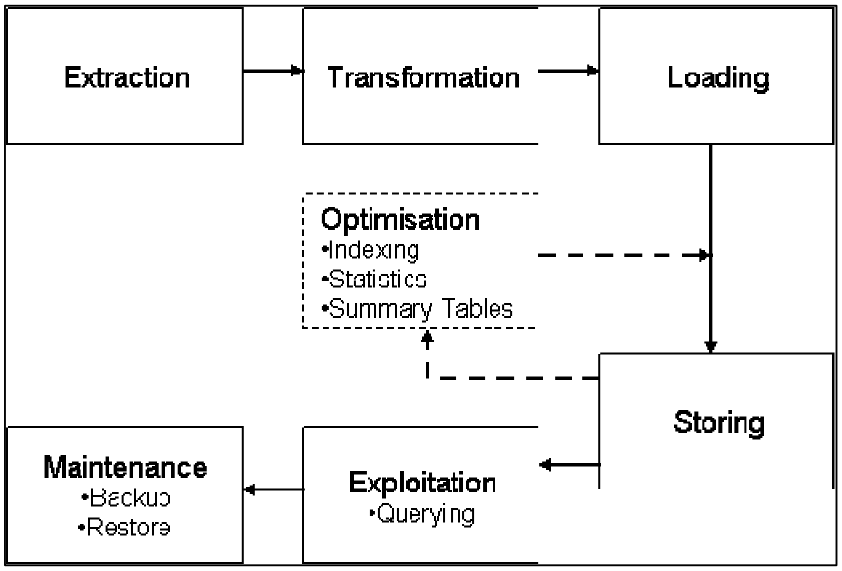


Figure 36 Phases of the Full Cycle of a Data Warehouse

6.3.2 Experimental Model

Once the phases were established (section 6.3.1), the architecture for the experiments was designed, as shown in Figure 37.

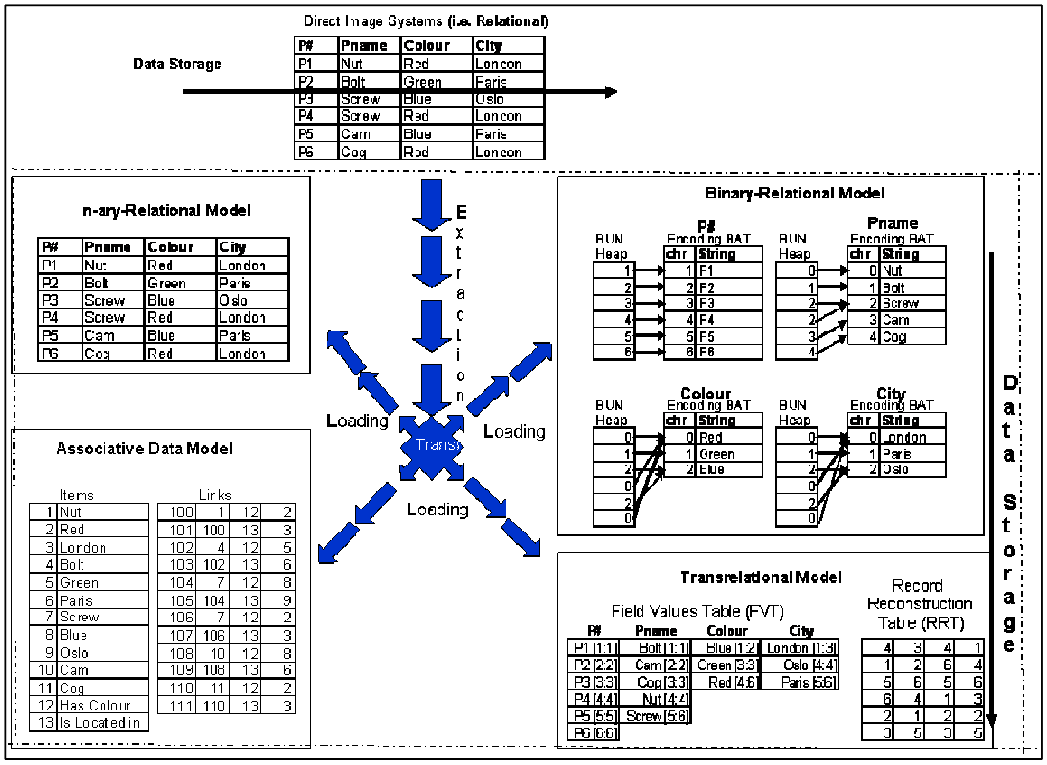


Figure 37 Experimental Model (Physical Storage Structures)

The experiments will consist of the following steps:

1. Data will be read and extracted from the horizontal repository.

2. Then data will be transformed according to the characteristics of the target data models.

3. The next step is to load the transformed data into the alternative data models instantiations, which follow different vertical data storage approaches.

4. Once the data are loaded in each data model instantiation, the extended TPC-H benchmark version (section 4.6) is executed against each DBMS.

5. Metrics measurement.

6.3.3 Experimental Metrics

The experimental metrics are those defined in the extended TPC-H benchmark (Table 6 and Table 7 in Chapter 4). The mapping of each metric to the corresponding phase of the DWH cycle is presented in Table 15 and Table 16.

Table 15 Extended TPC-H metrics in Pristine mode vs. DWH phases

Metric #	**Extended TPC-H metrics**	**DWH Phase**
1	Extraction times from transactional systems	Extraction
2	Transformation times to conform to the target data model structure	Transformation
3	Input file sizes measurement	Loading
4	Load data times into the Data Warehouse	Loading
5	Size of the Database tables after load (Database size)	Storing
6	Data Density Measurement	Storing
7	Query execution times (Pristine mode)	Exploitation
8	Data Warehouse Backup time	Maintenance
9	Backup size	Maintenance
10	Data Warehouse restore time	Maintenance

Table 16 Extended TPC-H technology specific metrics vs. DWH phases

Metric #	**Extended TPC-H metrics**	**DWH Phase**
11	Index creation times	Optimisation
12	Index sizes	Storing
13	Query times (with indexes)	Exploitation
14	Statistics computation times	Optimisation
15	Query times (with indexes & statistics)	Exploitation
16	Query times (with statistics without indexes)	Exploitation

The measurement of the metrics presented in Table 15 and Table 16 will allow the evaluation of the alternative data models and consequently the election of the best-suited data model for an Alternative Data Warehouse Architectural Configuration, as established in the research objective 4 in section 1.4.3.

6.4 Experimental Execution

In this section, the results of the experiments are presented. Those are grouped by their corresponding phase.

6.4.1 Data Extraction and Transformation phases

The logic behind the Extraction and Transformation phases is as follows:

a) Read data from the original data source (in this case Oracle RDBMS which uses a horizontal storage approach).

b) Extraction of data from the tables

c) Transformation of data

d) Write transformed data into flat files

In order to optimise the Extraction-Transformation and Loading (ETL) cycle, the transformation and reading were made at the same time. The transformation is concerned with manipulating data according to the structures required by each alternative data model. In general terms [Kimball, 1996], transformation includes a broader set of tasks, for example data type changes, transforming from the transactional database schema (Entity-Relationship) to a Data Warehouse schema (Star or Snowflake), data derivation (as summarisation, multiplication, truncation, etc.). In this case, it was not necessary to do these kinds of operations as the data source had the same database structure, that of the TPC-H (Figure 29).

Extraction and Transformation tools were developed for each of the alternative data models. They were implemented by SQL scripts; samples of these scripts are presented in Appendix A, section A.2.

One of the models to be considered as the target model during the ETL phases –and in the other phases - is Relational. This is because most of the current Data Warehouses read from Relational sources and write into Relational Data Warehouses. By considering Relational as another target, its behaviour within Data Warehouse environments has also been reported.

6.4.1.1 Data Extraction and Transformation Times (metrics 1 and 2)

Table 17 shows the Extraction and Transformation times with SF of 100MB.

Table 18 shows the Extraction and Transformation times with SF of 1GB.

Table 17 Extraction and Transformation times (SF=100MB)

TPC H Table Name	Number of Rows **(SF=100MB)**	Relational	Binary	Associative	Transrelational
Region	5	0.25	0.21	0.33	0.52
Nation	25	0.29	0.23	0.23	0.45
Supplier	1,000	0.47	0.35	0.37	0.74
Customer	15,000	2.78	2.76	3.23	2.94
Part	20,000	3.25	3.33	5.50	3.75
PartSupplier	80,000	12.40	13.36	13.81	14.10
Orders	150,000	23.60	25.01	26.17	26.76
Lineitem	600,572	109.44	108.56	114.83	117.54
	Total (Sec)	152.47	153.81	164.48	166.80
	(minutes)	2.54	2.56	2.74	2.78

Table 18 Extraction and Transformation times (SF=1GB)

TPC H Table Name	Number of Rows **(SF=1GB)**	Relational	Binary	Associative	Transrelational
Region	5	0.20	7.30	0.94	1.80
Nation	25	0.20	0.60	0.22	0.20
Supplier	10,000	41.30	3.10	2.00	2.30
Customer	150,000	2.00	27.40	27.14	28.50
Part	200,000	136.80	40.40	35.45	35.50
PartSupplier	800,000	25.70	137.20	135.21	154.60
Orders	1,500,000	249.10	251.10	252.51	280.70
Lineitem	6,001,215	1202.20	1301.00	1198.94	1221.30
	Total (sec)	1657.50	1768.10	1652.41	1724.90
	(minutes)	27.60	29.47	27.54	28.75

6.4.2 Loading Phase

Extracted data are written into flat files that are the input of the loading phase. The loading process was made using a “bulk load”, based on the corresponding load utility for each DBMS.

Owing to the massive workload involved during the initial Data warehouse load, it was made in batch mode. Therefore, in the experiment it was preferred to load in bulk to stress the system resources.

6.4.2.1 Input File sizes (metric 3)

The metric number 3 is the size of the files generated after the Extraction and Transformation processes (metrics 1 and 2). These input file sizes are the values of the files to be loaded into each data model instantiation, which are presented in Table 19 for SF=100MB and in Table 20 for SF=1GB.

Table 19 Input File Sizes (SF=100MB)

TPC H Table Name	Number of Rows **(SF=100MB)**	Relational	Binary	Associative	Transrelational
Region	5	0.00	0.00	0.00	0.00
Nation	25	0.00	0.00	0.00	0.00
Supplier	1,000	0.13	0.13	0.14	0.13
Customer	15,000	2.31	2.29	2.35	2.37
Part	20,000	2.29	2.26	2.30	2.37
PartSupplier	80,000	11.16	11.06	11.22	11.51
Orders	150,000	16.11	15.78	16.07	16.68
Lineitem	600,572	70.81	66.81	67.95	70.71
	Total	**102.81**	**98.35**	**100.03**	**103.77**

Table 20 Input File Sizes (SF=1GB)

TPC H Table Name	Number of Rows **(SF=1GB)**	Relational	Binary	Associative	Transrelational
Region	5	0.00	0.00	0.00	0.00
Nation	25	0.00	0.00	0.00	0.00
Supplier	10,000	1.34	1.33	1.36	1.38
Customer	150,000	23.23	23.05	23.60	23.94
Part	200,000	23.09	22.85	23.22	24.08
PartSupplier	800,000	113.31	112.35	113.87	117.59
Orders	1,500,000	163.95	160.73	163.58	171.11
Lineitem	6,001,215	724.66	684.64	696.08	729.36
	Total	**1049.58**	**1004.95**	**1021.71**	**1067.46**

As can be appreciated from Table 19 and Table 20, the Input file size differences are not big. Binary-Relational for both scale factors presented the smallest input file sizes. As there is less than 6% variation between the largest input files (Transrelational) and the smallest (Binary-Relational), the input file sizes can be considered constants.

6.4.2.2 Loading Data times (metric 4)

Following the experimental methodology and based on the full Data Warehouse cycle definition presented in 6.3.1, the next step is to load data into each data model instantiation and measure the required time to load data (metric 4).

Load times are presented in Table 21 and Table 22 for SF=100MB and SF=1GB.

Table 21 Load Times (SF=100MB)

TPC H Table Name	Number of Rows **(SF=100MB)**	Relational	Binary	Associative	Transrelational
Region	5	0.58	1.52	4.67	0.43
Nation	25	0.25	1.50	4.87	1.07
Supplier	1,000	0.47	1.51	24.24	0.99
Customer	15,000	7.12	1.89	424.59	6.02
Part	20,000	3.76	2.28	550.21	9.13
PartSupplier	80,000	13.67	2.90	2566.22	20.01
Orders	150,000	25.30	4.41	10964.01	64.26
Lineitem	600,572	91.31	16.61	93101.37	507.33
	Total (sec)	**142.46**	**32.61**	**107640.18**	**609.25**
	(minutes)	2.37	0.54	1,794.00	10.15

Table 22 Load Times (SF=1GB)

TPC H Table Name	Number of Rows **(SF=1GB)**	Relational	Binary	Associative	Transrelational
Region	5	0.30	1.50	3.90	0.40
Nation	25	0.10	1.50	4.40	1.31
Supplier	10,000	2.20	1.60	181.82	3.35
Customer	150,000	34.30	8.40	31800.00	59.84
Part	200,000	21.40	4.20	10200.00	97.69
PartSupplier	800,000	85.90	13.90	329940.00	249.47
Orders	1,500,000	174.70	26.50	4164173.00	855.05
Lineitem	6,001,215	873.50	246.00	1124536.00	10231.80
	Total (sec)	**1192.40**	**303.60**	**5660839.12**	**11498.91**
	(minutes)	19.87	5.06	94,347.32	191.65

In the case of the Associative/Triple Store model instantiation, with SF=1GB the load times for Orders and LineItem were estimated after loading just 300,000 records (Orders) and 10,000 records (LineItem) because their actual processing times were too long.

The best loading times were achieved for the binary-relational model instantiation, while the worst times were for associative/triple store instantiation, these being several orders of magnitude greater.

The best savings in time are achieved with the bigger tables. The binary-relational model instantiation requires between 74% and 79% less time loading the data set than the Relational Model instantiation, which is the closest one. Figure 38 presents the loading time savings for Binary-Relational when comparing it against Relational. The zero values correspond to the loading time for the relational model instantiation.

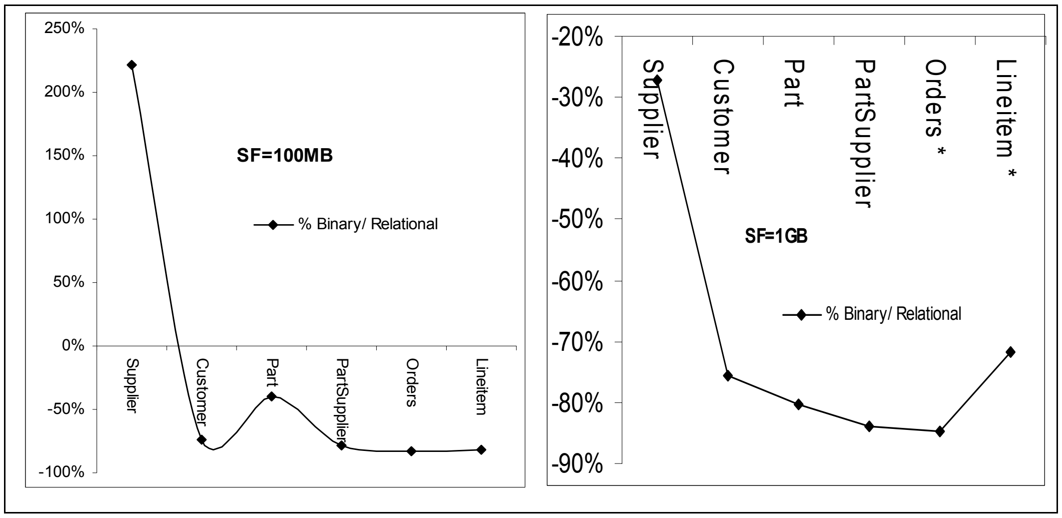


Figure 38 Loading Times Savings vs. Relational and vs. Binary

When comparing Binary-Relational versus Transrelational and Associative/Triple Store savings in loading time were even bigger: Binary-Relational has 99.99% savings against the Associative/Triple Store, which is the most dramatic comparison.

6.4.3 Storing Phase

The next phase of the experiment is the Storing phase where metrics 5 and 6 will be evaluated for each data model instantiation. Metric 6 (Data Density measurement) is one of the most important metrics as it will address research objective 1. The results for the Storing phase are presented in the following subsections.

6.4.3.1 Database Tables sizes (metric 5)

Once the data had been loaded, measurement of the database objects size was done. The results are presented in Table 23 and Table 24.

Table 23 Database Table size (SF=100MB)

TPC H Table Name	Number of Rows **(SF=100MB)**	Relational	Binary	Associative	Transrelational
Region	5	0.0625	0.0104	0.1875	0.0005
Nation	25	0.0625	0.0132	0.1875	0.0023
Supplier	1,000	0.1875	0.1839	1.5625	0.1600
Customer	15,000	3.00	2.69	25.63	2.79
Part	20,000	3.00	1.67	24.50	2.12
PartSupplier	80,000	13.00	11.41	72.81	13.05
Orders	150,000	19.00	13.07	196.50	18.56
Lineitem	600,572	88.00	54.46	1103.63	86.90
	Total (MB)	**126.31**	**83.52**	**1,425.00**	**123.58**

Table 24 Database Table size (SF=1GB)

TPC H Table Name	Number of Rows **(SF=1GB)**	Relational	Binary	Associative	Transrelational
Region	5	0.0625	0.0105	0.1875	0.0005
Nation	25	0.06	0.01	0.1875	0.002
Supplier	10,000	2.0	1.3	15.18	1.68
Customer	150,000	27.0	26.7	248.37	29.3
Part	200,000	30.0	16.3	247.25	21.52
PartSupplier	800,000	128.0	114.3	670.81	130.93
Orders	1,500,000	192.0	129.9	1,746.00	200.89
Lineitem	6,001,215	848.0	549.6	10,908.00	962.22
	Total (MB)	**1,227.1**	**838.1**	**13,835.99**	**1,346.54**

The total database size in the Binary-relational model instantiation has savings of 32% compared with the n-ary-relational model instantiation, no matter the scale factor used, demonstrating linear scalability. Bigger reductions are achieved with the bigger tables (*orders* and *lineitem*), whereas in contrast the Associative/Triple Store model produces bigger table sizes.

The Associative/Triple Store model database was 1,028% larger than the n-ary-relational. This is equivalent to having a Data warehouse 11 times larger than the same Data Warehouse stored in the n-ary-relational model.

It was found that part of this growth is caused by the way the Associative/Triple store model was implemented by SentencesDB, as it filled its pages at 56% to 59% and left

44% to 41% of empty space to be used by future database growth; this can be improved by future implementations of the associative/triple store model. It was not possible to modify the filling factor in SentencesDB, as it can be done in the n-ary-Relational DBMS [Koch, 1996].

In the case of the associative/triple store it was necessary to separate Customer and Supplier and re-name them 'c_name' and 's_name' in order to avoid violating implementation rule 1.5.2 [Williams, 2003]: "*The physical clustering of records within the database is allowed as long as this clustering does not alter the logical independence of each table*".

Another consideration was to create the Entity type Part_Supp to avoid violating implementation rule 1.5.2, even when in the Associative model it is not necessary to materialise this kind of association (Part-Supplier relation is required by the n-ary-Relational model), but it was created mainly in order to keep the eight entities (or tables) of the TPC-H (section 4.2) and to be able make a one-to-one comparison between each of the eight tables. By insisting on having eight tables in the Associative/Triple Store model it could be possible that the behaviour of the Associative/Triple Store was altered as it was forced to be modelled as if it was the n-ary-Relational. This can be considered as simulating n-ary-Relational with the Associative/Triple Store. Further investigation should be done in order to set a strictly Associative implementation of a TPC-H-like environment; but not forcing it to have eight tables and then observe and measure the final size of the Data Warehouse. However in the case of the current investigation, as it is more interested in the data density and data sparsity achieved by each model and not in the logical model aspect, then no further steps were taken to investigate the logical modelling aspect of the Associative/Triple Store model.

Another idea regarding the Associative/Triple Store logical database modelling is that it does not require the foreign keys to make the join of the records, and from here more disk savings can be achieved. This is interesting for Data warehouse environments, as a high percentage of the Data Warehouse size is to the result of foreign keys, mainly in the Fact tables. This characteristic is even more relevant in a Star schema (section 2.3.1.1) in which tables are not normalised. This aspect needs to be investigated by researchers interested in the logical modelling aspect of the databases, and in Data Warehouses logical modelling in particular.

It is important to describe the behaviour and considerations made during the data load of the Associative/Triple Store model instantiation. This will be made in the following paragraphs.

The initial size of the database (in SentencesDB it is named *chapter*) was 1,408 KB, and then the load of small tables (*Region, Nation, Supplier*) was made without any problem, but the *Part* table could not be loaded in one chunk, as *SentencesDB* ended with a *lack of memory* error message. This was for an SF=1GB.

In order to try to remedy this, the *Part* load file was split into chunks of 80,000 records by using the *UNIX split* command:

split -80000 PART_DATA_ASSOCIATIVE.csv PART_DATA_Associative

The original input file has 800,000 records. Apart from the split, the memory assigned to the *Java Virtual Machine* in SentencesDB was increased from 250MB to 384MB. This was made in the start configuration shell called *lazystart.sh*. With these changes, it was possible to load 80,000 records in 75 minutes, which was unacceptable when comparing against the reference model, in this case the n-ary-Relational model.

Many problems arose while trying to load the Associative/Triple Store instantiation and the following aspects were observed:

1. During the reading phase of a large flat file the CPU and the RAM worked well but,
2. During the commit phase of a large file the CPU usage grew to 100% and it was too slow to complete the commit.
3. It does not manage big files well because it tries to keep everything on RAM memory, and of course larger files do not fit into RAM.
4. The way to remediate the data loading was by splitting the input files into chunks of 10,000 records and leaving 256MB of RAM assigned to the Java Virtual machine. At least with these values it did not aborted, owing to the lack of memory. But other observations were made:

5. With the initial files it was faster but while more chunked-files were loaded the loading time increased from 6 minutes for the first file to 60 minutes for the last file. By observing this behaviour it was suspected that the DBMS is not releasing the RAM memory properly after loading each data chunk. This could be owning to the Java Virtual Machine itself or to the DBMS programming. Further investigation needs to be done by others interested in the optimisation of SentencesDB.
6. After 24 hours of loading data, the decision to stop it was made. An increment on the memory assigned to the Java Virtual machine was made assigning 300MB and splitting all files in chunks of 10,000 records.
7. When restarting with these parameters, the first file required 3 minutes to be loaded, versus the 60 minutes required by the latest 10,000 records chunks loaded before the process was stopped.
8. It was noticed that the size occupied by the loaded tables in the Associative/Triple Store model instantiation was huge in comparison with the n-ary-Relational model (the designated base model), and even worse than the Binary-Relational model. It was noticed that the block size used by SentencesDB was big (64KB) in comparison with the block size used by the n-ary-Relational (Oracle), which has an 8KB block size.
9. The block size was changed in SentencesDB to 8KB and the *Supplier* table was loaded. It was chosen because it has 10,000 records. It was loaded in 3 minutes and it occupied 15.18 MB within the database. It was noticed that the occupation percentage was low as it was reserving 44% of the space for future insertions into the database. It was not possible to modify the fill factor percentage in the DBMS *server properties*. Load times for three tables are presented in Table 25. In Table 26 database sizes and occupation percentages are presented.

Table 25 Associative/Triple Store Load times using two different block sizes

Table	Block Size=64K	Block Size=8K
Region	3.9 sec	3.87 sec
Nation	4.5 sec	4.5 sec
Supplier	208.59 sec	197 sec

Table 26 Associative/Triple Store database occupation percentage with two block sizes

Block Size (KB)	Number of Blocks	Total DB size (KB)	KB used	KB Free	Occupation %
64	271	17,344	9,757	7,587.48	56%
8	2,025	16,200	9,592	6608.47	59%

10. It can be observed from Table 25 and Table 26 that the total database size and the occupation percentage are not greatly improved by changing the block size. Now were the loading times improved. Based on these measurements, it was decided to do the data Load with the default block size = 64KB. The results are those already presented in Table 23 and Table 24.

The previously mentioned problems are related more to the SentencesDB implementation. No doubt its implementation can be improved, but optimisation of SentencesDB was not among the objectives of this research. From here Lazy Software (creator of SentencesDB) or others could create optimised versions of the Associative/Triple Store model after considering the aspects found in this research and reported in this thesis.

The Transrelational model for small Scale Factor (100MB) was 2% smaller than the n-ary-Relational model, but the larger Scale Factor (1GB) was 10% larger. This can be seen from Table 23 and in Table 24, from the same tables. It can be observed that the Binary-Relational model instantiation required less disk space than the rest of the models.

6.4.3.2 Data Density (metric 6)

In order to satisfy research objective 1, it was required to find which alternative data model has the highest Data Density. Metric 6 was evaluated, and the results are presented in Table 27 and Table 28.

Table 27 Data Density (SF=100MB)

TPC H Table Name	Number of Rows **(SF=100MB)**	Relational (rows/MB)	Binary (rows/MB)	Associative (rows/MB)	Transrelational (rows/MB)
Region	5	80	481	27	10,870
Nation	25	400	1,894	133	10,870
Supplier	1,000	5,333	5,438	640	6,250
Customer	15,000	5,000	5,570	585	5,376
Part	20,000	6,667	11,949	816	9,434
PartSupplier	80,000	6,154	7,009	1,099	6,130
Orders	150,000	7,895	11,475	763	8,082
Lineitem	600,572	6,825	11,028	544	6,911
	Total	6,861	10,376	608	7,012

Table 28 Data Density (SF=1GB)

TPC H Table Name	Number of Rows **(SF=1GB)**	Relational (rows/MB)	Binary (rows/MB)	Associative (rows/MB)	Transrelational (rows/MB)
Region	5	80	478	27	10,000
Nation	25	400	2,015	133	10,870
Supplier	10,000	5,000	7,568	659	5,952
Customer	150,000	5,556	5,619	604	5,119
Part	200,000	6,667	12,264	809	9,294
PartSupplier	800,000	6,250	7,001	1,193	6,110
Orders	1,500,000	7,813	11,543	859	7,467
Lineitem	6,001,215	7,077	10,920	550	6,237
	Total	7,058	10,334	626	6,432

Considering Table 27 and Table 28, Data Density varies depending on each table data, but in total terms, the alternative data model with the highest data density, no matter the Scale Factor, is the Binary-Relational Model. This model presented a data density equal to 10,000 rows/MB, while the Associative/Triple Store model presented the lowest data density, with only 600 rows/MB. The n-ary-Relational model instantiation has a data density of 7,000 rows/MB and Transrelational 6,500 rows/MB.

The Data Density of the *customer* table is nearly the same for n-ary-Relational, Transrelational, and Binary-Relational. Future work and study must be conducted on this to investigate the kind of data it has and understand why it does not vary its size.

6.4.4 Discarding some of the Alternative Data Models

Data Density is an important aspect that relates with the final size of the Data Warehouse. After measuring the data density of the data models, it was found that even while all of them avoid storing duplicates and abandon the horizontal approach to data management as established in theory (section 3.2) the data density for each data model is different. The research hypothesis "Data Density in Data Warehouse environments can be improved by using alternative Data Models" (section 1.4.2) is valid for the Binary-Relational Model; but not for the Associative/Triple Store nor for the Transrelational data model.
Another drawback of the Associative/Triple store was its Loading time as it was the slowest in this metric, with times of 19.9 minutes for the n-ary-Relational model vs. 94,347.3 minutes for the Associative/Triple Store using a Scale Factor=1GB. When investigating the causes of this poor performance, it was found that instantiation made by SentencesDB is programmed in Java and the performance was limited by the Java Virtual Machine. SentencesDB was run with different memory sizes for the virtual machine, but this did not make any difference (section 6.4.3). In contrast, the implementations of the other data models ran without modifying the real memory assignments.

For the Transrelational model, there were no big differences in Data Density or in the final Data Warehouse size. With the small scale factor of 100MB, it even produced smaller tables; in contrast, for the large scale factor (1GB), the Transrelational produced a larger Data warehouse than the n-ary-Relational instantiation. In any case, Transrelational never achieved a smaller Data Warehouse than the Binary-Relational instantiation (refer to Table 23 and Table 24). Hence the decision to go further with the Binary-Relational and not with the Transrelational was made.

Other important factor is that, as far as could be determined, during this thesis the first public domain implementation of the storage mechanism for the Transrelational Model was made (refer to Chapter 5) and which was reported in [Gonzalez, 2006a]. It could be possible to make improvements to this implementation, but it was not one of the main objectives of this research to full develop and optimize a Transrelational model instantiation. Only the essential storage algorithms were implemented to prove whether it helps or not to manage large databases, in this case within the context of Data Warehousing.

Another factor regarding transrelational is that the algorithms (section 5.3) involved to achieve the elimination of duplicates at the column level are more complex than those involved in the binary-relational. Those algorithms are high processing consumers and do not report major benefits in time when compared with n-ary-Relational and with binary-relational.

For the reasons mentioned before, it was decided to stop testing with Transrelational and with Associative/Triple Store, and to continue the investigation with the n-ary-Relational model and the Binary-Relational model, which have major improving differences against the base model (n-ary-Relational).

6.4.5 Exploitation Phase

Long Query execution times are the main complaint of end users [Pendse, 2003a] and that is a manifestation of a suboptimal Data Warehouse Environment. The following section presents the results of measuring the query execution times for the remaining data models.

6.4.5.1 Queries Execution times in pristine mode (metric 7)

Query Execution Times in *Pristine mode* are shown in Table 29. As defined earlier in section 4.6, *Pristine mode* refers to loading data into the instantiations of the n-ary-Relational and Binary-relational models, and executes the queries against this data set without running indexation or statistics computation, which have great influence over the RDBMS Query Optimizer [Date, 2004], but these are not part of the relational model [Codd, 1990].

The query language used was SQL, even though MonetDB offers a native query language called MIL [MonetDB, 2004b] that could thus avoid the translation time from SQL language to MIL. However, even with this translation the query response times were superior.

On Table 29 with SF=100MB for the Binary-Relational model, all queries ran in sub-seconds with the exception of 3 queries, while in n-ary-Relational only 3 queries ran in sub-seconds and query 4 needs 31.35 minutes to run.

With SF=1GB, all the queries in the Binary-Relational model ran in seconds (none of them reached 1 minute), while queries in n-ary-Relational ran in minutes. The worst cases were Query 4 with 14 days and Query 21 with 30 days.

Table 29 Query Execution Times in Pristine Mode

SF=100MB	Relational (seconds)	Binary (seconds)
Query 1	4.66	1.46
Query 2	0.51	0.31
Query 3	2.22	0.25
Query 4	1887.49	0.23
Query 5	2.59	0.19
Query 6	0.93	0.17
Query 7	4.86	0.42
Query 8	2.61	0.16
Query 9	2.71	0.67
Query 10	4.71	0.16
Query 11	1.18	0.12
Query 12	1.58	0.41
Query 13	4.01	1.33
Query 14	1.09	0.15
Query 15	1.96	0.15
Query 16	0.84	0.22
Query 17	16.15	0.16
Query 18	8.12	0.56
Query 19	2.69	0.66
Query 20	196.49	0.26
Query 21	13.06	1.33
Query 22	10.79	0.12
Total time (secs)	2171.24	9.49
In minutes	**36.19**	**0.16**

SF=1GB	Relational (seconds)	Binary (seconds)
Query 1	240.03	18.94
Query 2	451.08	2.80
Query 3	129.38	11.58
Query 4	1195478.31	13.25
Query 5	240.50	15.64
Query 6	48.20	4.31
Query 7	304.74	16.62
Query 8	143.72	8.68
Query 9	2453.11	16.62
Query 10	108.54	8.68
Query 11	29.45	1.93
Query 12	66.70	8.28
Query 13	43.80	30.93
Query 14	50.18	6.59
Query 15	94.67	7.12
Query 16	1262.90	4.68
Query 17	9757.58	6.50
Query 18	640.84	7.08
Query 19	146.06	10.33
Query 20	19087.81	6.09
Query 21	2549012.53	17.24
Query 22	9371.32	4.79
Total time (secs)	3789161.43	228.70
in minutes	63152.69	**3.81**
in hours	1052.54	
in days	**43.86**	

The Total Query workload is the total processing time for the 22 queries of the TPC-H benchmark. As can be appreciated from Figure 39, the queries' execution time differences are considerable. With SF=100 MB, Binary-Relational required 9.5 seconds to process the 22 queries, while n-ary-Relational required 36.1 minutes. With SF=1GB, Binary-Relational took 3.8 minutes to process the 22 queries, while n-ary-Relational took 43.8 days.

These are the times using only relations (tables) as defined by the relational model, and it is the way to compare model achievements, but of course in order to have pragmatic results, the measurement of the queries uses extended relational technology (indexes and statistics only) that are used extensively in relational products. These results are presented in section 6.4.7.

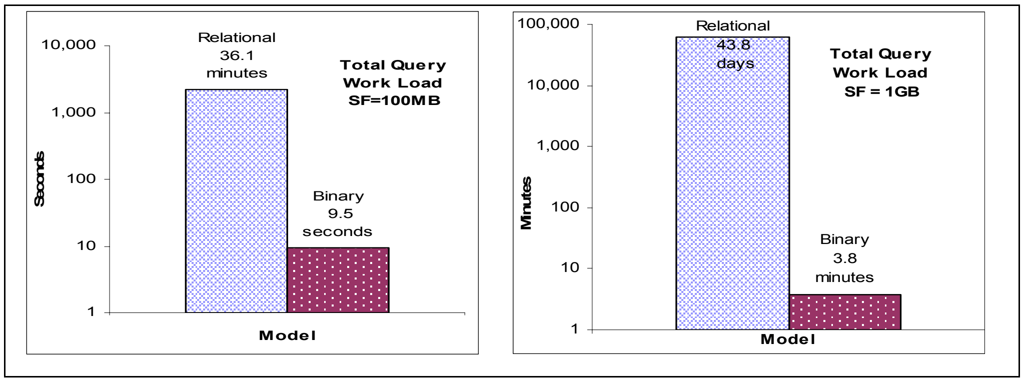


Figure 39 Total Query Work Load in Pristine Mode

6.4.6 Maintenance Phase

Every system requires a maintenance phase. In this phase the minimum operations required are to Backup data and if necessary to Restore data. Those aspects are considered in this section.

6.4.6.1 Data Warehouse Backup time (metric 8)

Metric 8 is the time required to make a Backup of the Data Warehouse. The Binary-Relational DBMS required less time to do the backup, no matter the scale factor; time savings are considerable as the Binary-Relational DBMS achieved savings between 72 to 82% when compared with the n-ary-Relational DBMS. In Table 30, the results of time and savings aspects are presented.

	Scale Factor = 100MB	
	Backup time (seconds)	**% Savings (vs. Relational)**
Relational	34.76	0.00%
Binary	6.16	**82.29%**

	Scale Factor = 1GB	
	Backup time (seconds)	**% Savings (vs. Relational)**
Relational	400.61	0.00%
Binary	111.74	**72.11%**

Table 30 Backup time and savings percentages for n-ary-Relational and Binary-Relational

6.4.6.2 Backup size (metric 9)

The other aspect to be considered when backups are made is their size. Metric 9 has been defined in order to reflect this aspect of the maintenance phase of a Data Warehouse operation. In Table 31, the Backup and Database sizes for n-ary-Relational and Binary-Relational are presented.

	Scale Factor = 100MB	
	DB Size (MB)	Backup Size (MB)
Relational	126.31	114.00
Binary	83.52	83.64

	Scale Factor = 1GB	
	DB Size (MB)	Backup Size (MB)
Relational	1,227	1,157
Binary	838	838

Table 31 Backup size vs. Database size for n-ary-Relational and Binary-Relational

This can appear trivial, but as the Binary-Relational database is smaller than the n-ary-Relational database, the backups for the Binary-Relational are also smaller; this is consequence of the Binary-Relational highest Data Density.

6.4.6.3 Restore time (metric 10)

The other aspect considered within the maintenance phase is the Restore operation once the Backup has been made. Restoring times are show in Table 32.

	Scale Factor = 100MB	
	Restore time (seconds)	**% Savings (vs. Relational)**
Relational	78.92	0.00%
Binary	10.42	**86.79%**

	Scale Factor = 1GB	
	Restore time (seconds)	**% Savings (vs. Relational)**
Relational	504.38	0.00%
Binary	103.06	**79.57%**

Table 32 Restore times and savings for n-ary-Relational and Binary-Relational

6.4.7 Technology Specific Metrics to Improve Query Performance (metrics 11 to 16)

Technology specific metrics were utilised to produce pragmatic results for the n-ary-relational model. One way to improve performance on relational products is by indexing the tables and computing statistics, but these activities need extra processing time and disk space. A set of metrics (metric 11 to metric 16) was defined in section 6.3.3, and they will measure technology-specific metrics. Figure 40 shows the total processing time in the Relational model instantiation, which includes Data loading, Index creation and statistics computation.

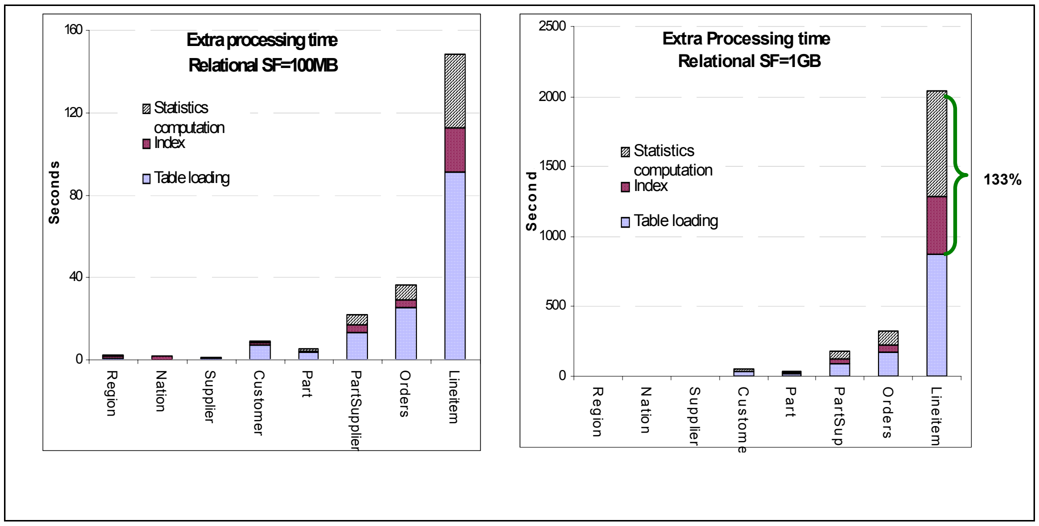


Figure 40 Extra Processing Time to Improve Performance in Relational

From Figure 40 it can be appreciated that the indexing and statistics computation times are not small when compared with the loading time required, so those times deserve to be considered during the optimisation phase of Relational DBMS. Those times have more relevance in larger tables.

In Table 33 and Table 34, the indexes sizes required by Relational are presented, and, as can be appreciated for large tables (*orders* and *lineitem*), the size of the index is between 27% and 44% of the table size itself. Hence indexes are not a trivial aspect of the query performance optimisation, as they require considerable disk space. The total extra disk space required by the indexes is 40% of the table sizes, which is not a small amount of disk.

Table 33 Indexes Space Requirements by n-ary-Relational (SF=100MB)

Relational Indexes Sizes (SF=100MB)	**Table Size**	PK	FK1	FK2	**Total Indexes Size**	% Indexes Size vs. Table Size	**Total Size (Table+ Indexes)**
	(MB)	(MB)	(MB)	(MB)	(MB)		(MB)
Region	**0.0625**	0.0625			**0.0625**	100.00%	**0.13**
Nation	**0.0625**	0.0625	0.0625		**0.125**	200.00%	**0.19**
Supplier	**0.1875**	0.0625	0.0625		**0.125**	66.67%	**0.31**
Customer	**3.00**	0.25	0.25		**0.5**	16.67%	**3.50**
Part	**3.00**	0.375			**0.375**	12.50%	**3.38**
PartSupplier	**13.00**	2	2	2	**6**	46.15%	**19.00**
Orders	**19.00**	3	3		**6**	31.58%	**25.00**
Lineitem	**88.00**	12	11	13	**36**	40.91%	**124.00**
	126.31			**Totals**	**49.2**	**38.94%**	**175.50**

Table 34 Indexes Space Requirements by n-ary-Relational (SF=1GB)

Relational Indexes Sizes (SF=1GB)	**Table Size**	PK	FK1	FK2	Total Indexes Size	% Indexes Sizes vs. Table size	**Total Size (Table+ Indexes)**
	(MB)	(MB)	(MB)	(MB)	(MB)		(MB)
Region	**0.0625**	0.0625			**0.0625**	100.00%	**0.13**
Nation	**0.06**	0.0625	0.0625		**0.125**	200.00%	**0.19**
Supplier	**2.0**	0.1875	0.1875		**0.375**	18.75%	**2.38**
Customer	**27.0**	3	3		**6**	22.22%	**33.00**
Part	**30.0**	4			**4**	13.33%	**34.00**
PartSupplier	**128.0**	17	14	14	**45**	35.16%	**173.00**
Orders	**192.0**	26	27		**53**	27.60%	**245.00**
Lineitem	**848.0**	128	112	136	**376**	44.34%	**1,224.00**
	1,227.1			**Totals**	**484.6**	**39.49%**	**1,711.69**

After creating indexes and computing statistics in order to help the query optimizer to produce better execution plans, three scenarios were run:

1. Queries executed only with indexes
2. Queries executed only with statistics
3. Queries executed with both indexes and statistics

The results are presented in Figure 41, where the Binary-Relational results have also been included for comparison purposes.

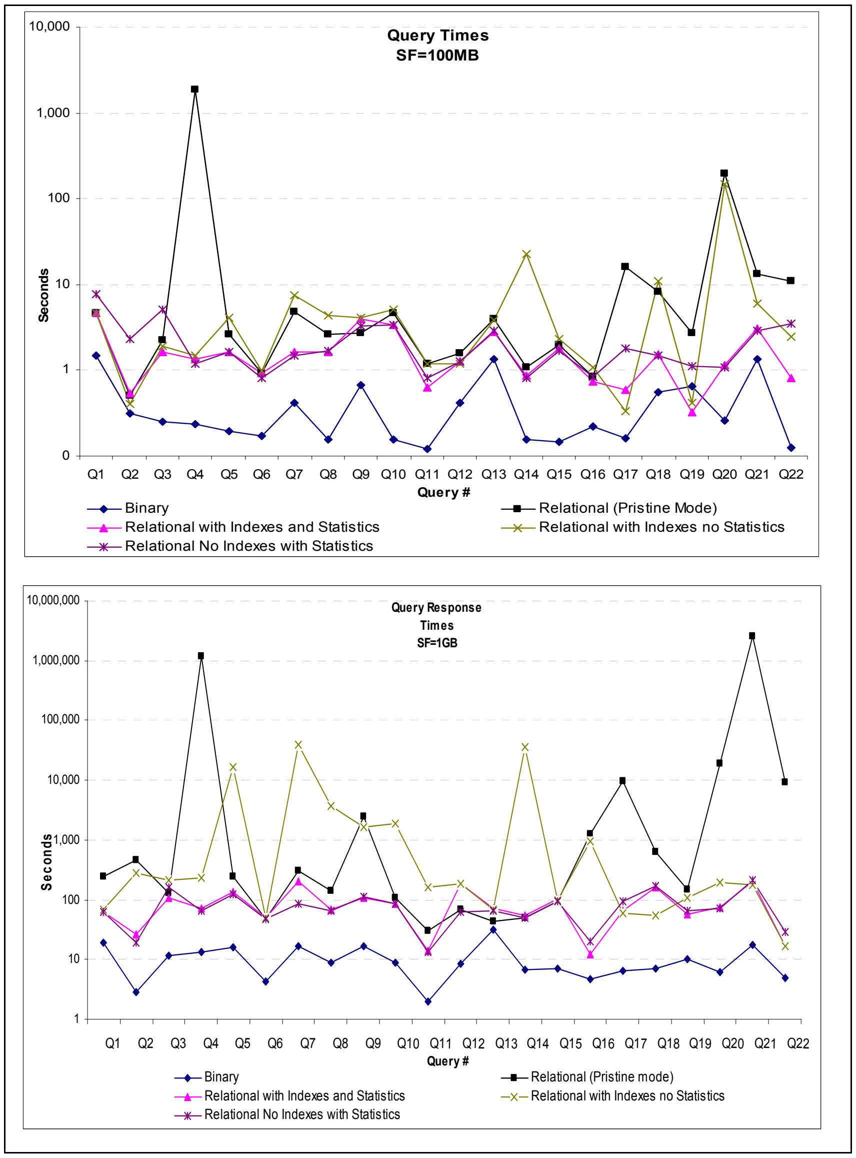


Figure 41 Query Response time with different Scenarios to Optimise n-ary-Relational

As can be appreciated in the case of SF=100MB, only one query ran faster in the n-ary-Relational than in the Binary-Relational, but with SF=1GB not a single query ran faster in n-ary-Relational than in Binary-Relational.

6.5 Conclusions

The experiments carried out with the different models have shown that the Binary-Relational model has the highest data density and performs faster loads; at the same time it reduces the disk space and increases query performance, apart from improving the backup and restore times. All these characteristics are important in Data Warehouse environments and they are good solutions to the problems that organisations are facing and which were the motivations of this research.

Fast data loads are important, as ETL times are time-consuming tasks which require a lot of night time to be completed by the organisations. Therefore, if a reduction on the processing time is feasible then organisations will be positively affected.

Because of its better data density, the Binary-Relational Data Warehouse was smaller than the Data Warehouses of the other data models. This is beneficial for organisations, as they will spend less money in disk storage.

Even more important Queries are faster while executed against a Binary-Relational repository. This is because of its vertical approach to manage data. Further analysis of these results will be made in Chapter 7.

CHAPTER 7
Analytical Model and Data Analysis

7.1 Introduction

In this chapter, the analysis of the data obtained during the experiments presented in chapter 6 is made. In order to analyse data obtained from the different data models, an analytical model is proposed, during the analysis the full data warehouse cycle as defined in section 6.3.1 will be followed and data will be analysed phase by phase.

In this chapter the conclusion relating to which is the best data model to be adopted in Data Warehouses is made.

7.2 Analytical model

The experiment measurements have been made on the previous chapter, but in order to make the analysis, an analytical model is created.

Consider a Relational Database with the following characteristics:

a) No duplicates
b) No missing values

Where:
S_R = Size of Relational
n = number of columns
r = number of rows

Convert to a Binary-Relational (BR) Database:
It is necessary to add the key to every column, except the primary key itself.

$$S_{BR} = S_R + (n-1) * r * KeySize$$

For each duplicated or missing value, it is necessary to reduce the size of the value itself but at the same time add an Object Identifier (OID) which will avoid losing the way of reconstruct the original record [Boncz, 2002]:

$$S_{BR} = S_{BR} - \text{SizeOfValue} - \text{KeySize} + \text{OID}$$

In the case of the Transrelational (TR) Database, it is necessary to consider the size of the rrt table

$$S_{TR} = S_R + (n * r * \text{IntegerSize})$$

For each duplicated or missing value, it is necessary to decrease the size of the value, but in this case the integer size remains as it is part of the *record reconstruction table* (rrt).

For each duplicated or missing value, it is necessary to decrease the size of the value but add the size which corresponds to the row ranges [Date, 2004], which are integer values which indicate the beginning and the end of the rows which relate to a certain value.

$$S_{TR} = S_{TR} - \text{SizeOfValue} + (2*\text{IntegerValue})$$

Hence the size of the Transrelational can be modelled in the following terms:

$$S_{TR} = S_{BR} - (r*\text{IntegerSize}) - (\text{OID}*n*r) + (n*r*\text{IntegerSize}) + (2*\text{IntegerSize}*\#\text{UniqValues}) - (r*2*\text{IntegerValue})$$

Where:

SBR is the Size of the Binary Relational database without duplicates.

*(r*IntegerSize)* is the size of the PK which was not compressed on the Binary-Relational, but its size should be considered on the Transrelational for the rrt table size.

*(OID*n*r)* is the space occupied by the OIDs which are used on the Binary-Relational Model, but not on the Transrelational.

(n r*IntegerSize)* is the size of the rrt table.

*(2*IntegerSize*#UniqValues)* is the size of the row ranges kept on the fvt table.

*(r*2*IntegerValue)* is the size of the PK which is not compressed and does not generate row ranges.

Working with the formula it can be re-write as:

$$S_{TR} = S_{BR}-(r\ (IntegerSize+(OID*n)+(2*IntegerSize)) + (IntegerSize((n*r)+(2*\#UniqValues)))$$

If IntegerSize = 8 bytes and OID = 2 bytes then:

$$S_{TR} = S_{BR} - 24r + 6nr + 16*UniqValues$$

Where

24r is the size of the primary key and its corresponding row ranges.

6nr is the size of the rrt table size minus the total OIDs size.

*16*UniqValues* is the size of the row ranges.

The extra space used by the rrt table plus the extra space used by the row ranges of the fvt table are the elements which made the Transrelational database larger than the Binary-Relational database.

This analytical model will be used to analyse the results obtained on Chapter 6 comparison of the theoretical results are contrasted against the real measurements for the models instantiations.

7.3 Experiment Data Analysis

Experimental data results were presented in Chapter 6. Now the analysis of such data is made in the following subsections.

7.3.1 Extraction and Transformation Phases

Two metrics are considered in these phases:

- Metric 1: “Extraction times from Transactional Systems” and
- Metric 2: “Transformation times to conform to the target data model structure”

From the results presented in Table 17 and Table 18 (section 6.4.1.1), it has been concluded that Extraction and in this case, Transformation times can be considered constants, because time differences between the fastest and the slowest data model instantiations were small (0.3 minutes or 9.3% for SF=100MB and 2 minutes or 7.2% for SF=1GB).

This is because Extraction time depends on the speed to read and return data of the source system; this is a constant element for all data models instantiations as the source system is the same. The overhead required by the transformation is similar as transformations are really simple; thus the Extraction and Transformation times can be considered constants, regardless of the target alternative data model.

7.3.2 Loading Phase

A fast data load is important because it is a time-consuming task and organisations are suffering with their data warehouse loading [Feinberg, 2007]. Usually this is one of the tasks involved in the night processing time, but now with the birth of the second generation of the Business Intelligence (BI 2.0) [Raden, 2007] where near-real time loading is needed faster loads are essential [Feinberg, 2007] to enable data loads to be done more frequently.

Since the beginning of the Extraction-Transformation and Loading (ETL) concepts [Devlin, 1988] this phase has followed the sequence: data Extraction, then data Transformation and finally data Loading. When following the ETL approach, data are Extracted, Transformed and Loaded in sequential fashion and on a row-by-row basis.

Some examples of commercial tools that follow this approach are referred to at Informatica [2007], Sybase [2007], and IBM [2008b]. They read one row which is transformed and then inserted into the Data Warehouse. Figure 42 is an example of such architecture. In this architecture, Transformation is made in a middle tier computer, usually a smaller computer than the one of the source and target systems, consequently with less processing power and with slower disks, hence constituting a bottle neck on the whole Data Warehouse cycle [Nicholas, 2006].

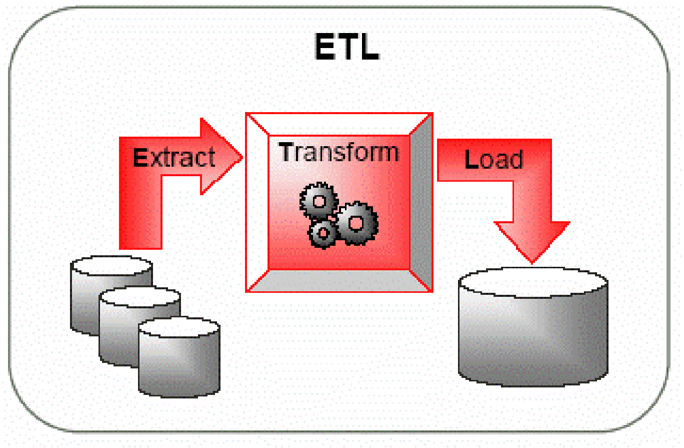


Figure 42 An ETL architecture. From Nicholas [2006]

Another approach can be followed which changes the ordering of tasks, following an Extraction-Loading and Transformation (ELT) approach [Nicholas, 2006]. This approach has advantages because data is Extracted and Loaded in to the Data Warehouse machine, which usually has more computer power and might include parallel architectures such as Teradata [2007] or Netezza [2008], and then the Transformation is made on the powerful computer.

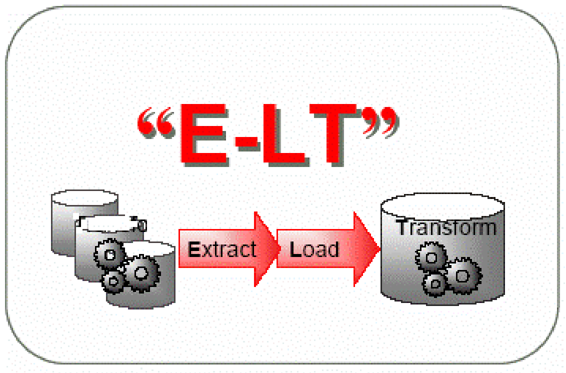


Figure 43 An ELT architecture. From Nicholas [2006]

With the ELT approach, data are loaded in bulk instead of doing row-by-row insertions as in the ETL approach. The ELT approach was used in the experiment. It was preferred to load in bulk to stress the system capabilities. This bulk load is similar to the initial Data Warehouse Load.

As mentioned before, faster loads are demanded by the industry and vendors have improved their loader technology to do parallel loads [Corey, 1997] to try to remedy the loading times. This approach usually parallelizes groups of rows that are sent to each loading thread. In contrast, the Binary-Relational follows a vertical approach which enables the parallel loading by columns. These behaviours are depicted in Figure 44.

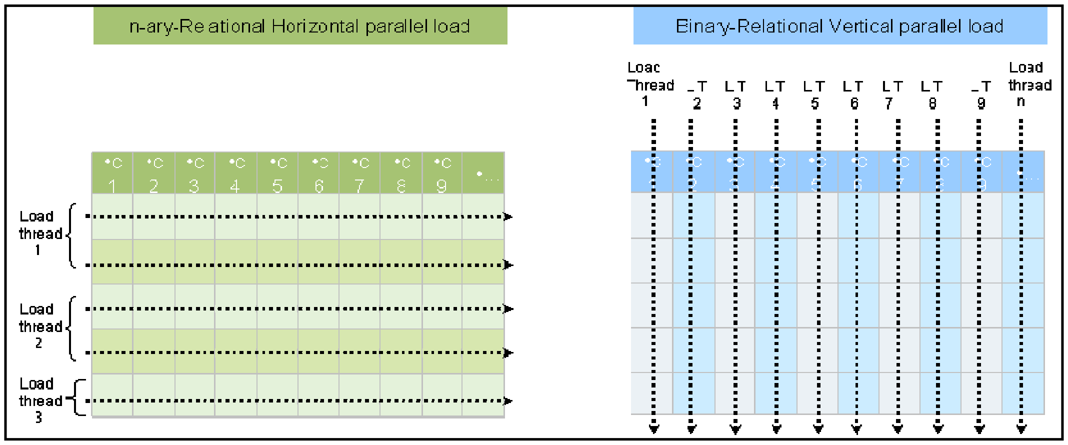


Figure 44 Horizontal vs. Vertical parallel loads

As can be observed in Figure 45 and in Figure 46 (SF=1GB), the faster model in the *Loading* phase is the Binary-Relational model. These loading times were achieved by the Binary-Relational model, basically because it can load each row in parallel. More detail on how the Binary-Relational model favours the row-parallel loading is described in section 7.3.2.1. By contrast relational implementations can not do the parallelisation at column level, instead they send groups of rows to be processed by different processing threads and the parallelisation is made by groups of rows; but the performance is lower.

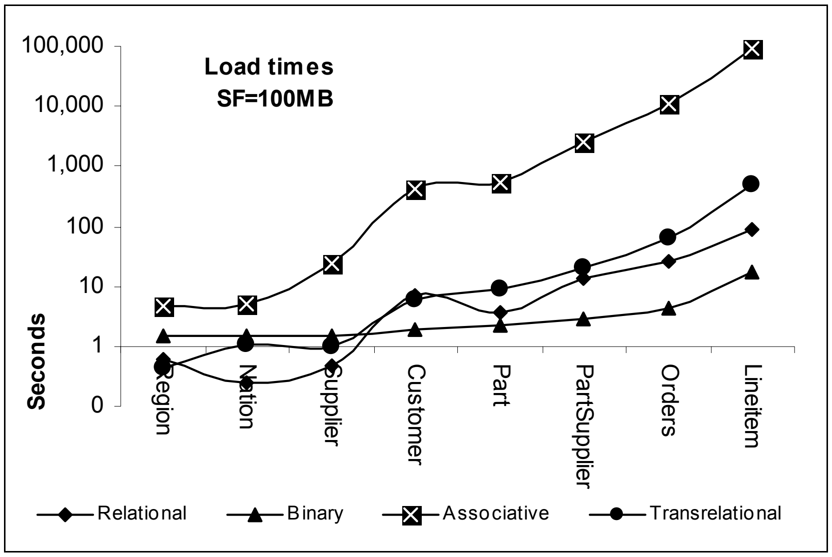


Figure 45 Load Times (SF=100MB)

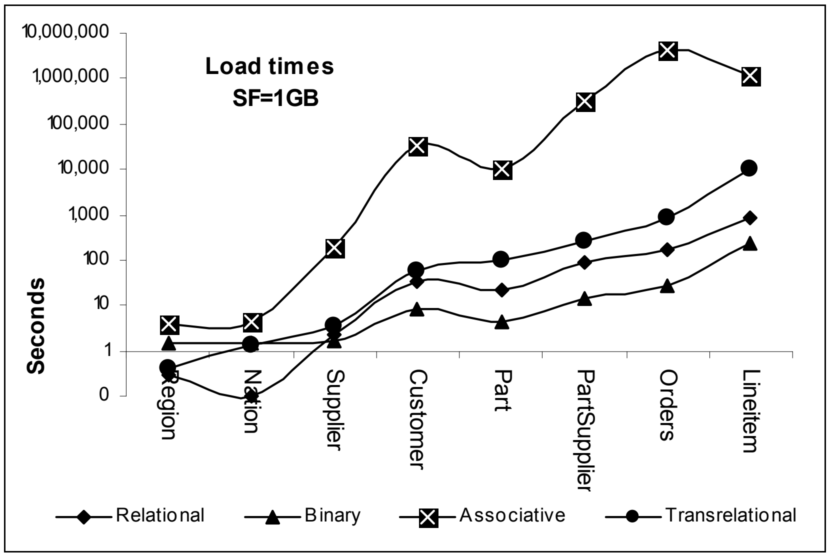


Figure 46 Load Times (SF=1GB)

7.3.2.1 Binary-Relational Load Process

In Binary-Relational each column is an independent relation of degree 2; the loading of data can be done in parallel by starting many loading threads, one for each column of the table. This process is detailed in the following paragraphs.

For each table two things are determined: first the degree of the relation (number of columns) and then it is converted to n relations of degree=2 (Binary relations); thus during the Load phase a processing thread can be started for each binary relation (column). The result is that the table is loaded in parallel; each column is processed independently of each other.

Also for each binary relation (column), the cardinality is determined (number of different values) in order to create the hash function considering n-buckets (where n=distinct values which equals the cardinality of the column), and, according to section 3.4, values are written just once and complementary storage structures are kept to avoid losing which rows are sharing each one of the distinct values. These complementary structures are called Binary Association Tables (BAT) [Kersten, 2005] as described in section 3.4.2. The whole process is illustrated in Figure 47.

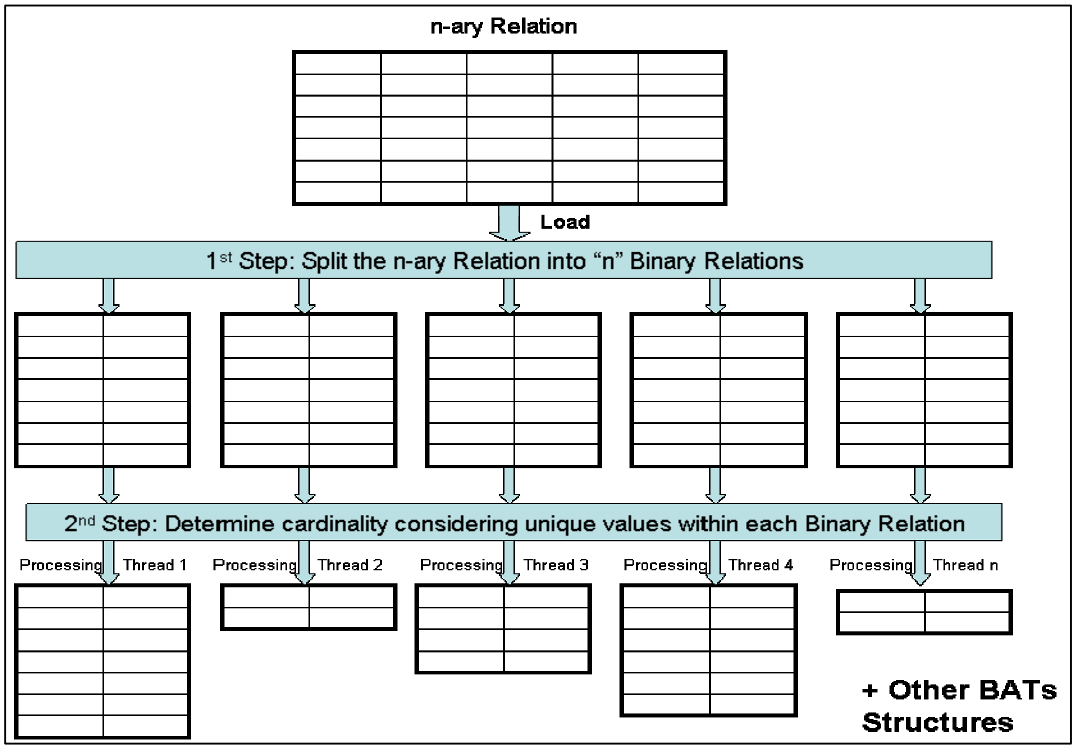

n-ary Relation
Load
1st Step: Split the n-ary Relation into "n" Binary Relations
2nd Step: Determine cardinality considering unique values within each Binary Relation
Processing Thread 1
Processing Thread 2
Processing Thread 3
Processing Thread 4
Processing Thread n
+ Other BATs Structures

Figure 47 Process of Loading into Binary-Relational Model Instantiation

7.3.3 Storing Phase

Storing is directly related to the size of the tables within the database and finally with the data density.

7.3.3.1 Database Tables sizes

When sizing databases, it is common to add 20% to 30% additional space of that spaced occupied by the raw data. With these results, the Binary-Relational model challenges the assumption that data will grow once loaded into a database [Haughey, 2006], as it achieved savings of 32% compared to the n-ary Relational model (see Table 23 and Table 24 in Chapter 6).

Now it is necessary to compare the experimental results against the theoretical results from the analytical model. These detailed results are presented in Table 35 and in Table 36.

Table 35 Detailed Results of the Analytical Model vs. Experimental Results for SF=100MB

SF=100 MB	Columns	Rows	# UniqVal	n-ary-Relational (Theory)	n-ary-Relational (real)	Binary-Relational (Theory)	Binary (Real)	Transrelational (Theory)	Transrelational (Theory)	Transrelational (Real)	OID=	2	Int=	8
Table	(n)	[r]		(MB)	(MB)	(MB)	(MB)	(Bytes)	(MB)	(MB)		2		8
Region	3	5	15	0.001	0.063	0.00	0.01	1,245	0.00	0.0005		2		8
Nation	4	25	80	0.005	0.063	0.01	0.01	6,585	0.01	0.0023		2		8
Supplier	7	1,000	6,024	0.20	0.19	0.25	0.18	372,761	0.36	0.16		2		8
Customer	8	15,000	89,912	3.30	3.00	3.85	2.69	5,830,858	5.56	2.79		2		8
Part	9	20,000	52,475	3.41	3.00	2.04	1.67	3,581,848	3.42	2.12		2		8
PartSupplier	5	80,000	165,979	18.16	13.00	17.62	11.41	21,616,209	20.61	13.05		2		8
Orders	9	150,000	457,750	22.46	19.00	19.22	13.07	31,980,639	30.50	18.56		2		8
Lineitem	16	600,572	695,255	112.83	88.00	38.47	54.46	94,709,105	90.32	86.9		2		8

Table 36 Detailed Results of the Analytical Model vs. Experimental Results for SF=1GB

SF=1 GB	Columns	Rows	# UniqVal	n-ary-Relational (Theory)	n-ary-Relational (Real)	Binary-Relational (Theory)	Binary (Real)	Transrelational (Theory)	Transrelational (Theory)	Transrelational (Real)	OID=	2	Int=	8
Table	(n)	[r]		(MB)	(MB)	(MB)	(MB)	(Bytes)	(MB)	(MB)		2		8
Region	3	5	15	0.001	0.063	0.00	0.01	1,245	0.00	0.0005		2		8
Nation	4	25	80	0.005	0.060	0.01	0.01	6,585	0.01	0.002		2		8
Supplier	7	10,000	59,979	2.02	2.00	2.46	1.30	3,718,920	3.55	1.68		2		8
Customer	8	150,000	890,077	33.04	27.00	38.25	26.70	57,948,523	55.26	29.3		2		8
Part	9	200,000	467,413	34.14	30.00	18.81	16.30	33,199,806	31.66	21.52		2		8
PartSupplier	5	800,000	1,119,864	181.58	128.00	165.00	114.30	195,734,703	186.67	130.93		2		8
Orders	9	1,500,000	4,425,458	223.16	192.00	183.53	129.90	308,253,017	293.97	200.89		2		8
Lineitem	16	6,001,215	5,493,016	1,110.30	848.00	324.96	549.60	860,716,615	820.84	962.22		2		8

Table 37 Analytical Model vs. Experimental Results

SF=100MB	n-ary-Relational	Binary-Relational	Transrelational	Diff. Transrel - Binary (%)	Savings (Binary vs. Relational)	Savings (Transrelational vs. Relational)
Theory	160.38	81.46	150.78	46%	49%	6%
Real	126.31	83.51	123.58	32%	34%	2%
Difference (%)	21%	-3%	18%			

Table 38 Analytical Model vs. Experimental Results

SF=1GB	n-ary-Relational	Binary-Relational	Transrelational	Diff. Transrel - Binary (%)	Savings (Binary vs. Relational)	Savings (Transrelational vs. Relational)
Theory	1,584.25	733.01	1,391.96	47%	54%	12%
Real	1,227.12	838.12	1346.54	38%	32%	-10%
Difference (%)	23%	-14%	3%			

From Table 37 and Table 38 it can be observed that the n-ary-Relational database is smaller compared to the expected size obtained from the theoretical size. It is because the technology that has been implemented in the RDBMS, in this case Oracle, in the case of character strings, they have been declared as *varchar* data type, which avoid storing the full size of the field, it only stores the actual size of the string [Oracle, 2005] in order to save space. On the actual TPC-H database schema there are many char fields which have been declared as varchar. The main space reduction is because of the comment field which nearly all tables have. The actual size difference between the theory and the implementation is 21%.

Keeping on the theoretical sizes, the Binary-Relational should save around 49% and 54% of the space when compared against the n-ary-Relational model; but in practice, according to the measurements of the experiment, the Binary-Relational implementation achieved space savings between 32 and 38%. The important fact is that the Binary-Relational model achieves considerable space savings when compared to the n-ary-Relational model both in practice and in theory.

The difference between the Binary-Relational analytical model and the practical implementation made in MonetDB is between 3% and 14% larger on the implementation.

Analysing the Transrelational Model and its implementation, the following conclusions can be made:

According to the analytical model, the Transrelational model could achieve space savings between 6 and 12% depending of the database size, the larger the database the larger the savings are. But in reality on the implementation made of the storage mechanism for the Transrelational, only achieved savings of 2% for the small scale factor (instead of the 6% expected) and for the larger scale factor, the Transrelational implementation instead of decrease, it grew 10% respect of the n-ary-Relational. Anyway this small size reduction could not justify a change from the existing n-ary-Relational model.

When analysing the theoretical size and comparing to the implementation made of the storage mechanism for the Transrelational, the actual database size achieved with the implementation is smaller between 3 and 18% than the theoretical size, this is because the improvement introduced to the fvt table column condensation process algorithm in 5.4.1, which has been called the *condensation threshold*. This threshold avoids condensing columns (duplicates elimination) when the condensation process will produce a larger column size because of the size required by the row ranges (refer to Chapter 5). The analytical model considers that all the columns on the fvt are condensed, but the selective condensation implemented on version 2.3 of the storage mechanism for the Transrelational model does not condense all columns (refer to Chapter 5). The positive effect of the selective condensation and the condensation threshold can be observed on the sizes achieved by individual tables of the TPC-H schema (refer to Table 35 and Table 36) but also this difference has been measured when different implementations of the storage mechanism for the Transrelational model were made (refer to section 5.5)

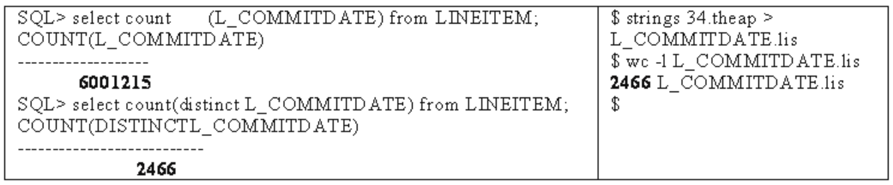
Analysing the size difference between the Binary-Relational and the Transrelational models, even when both eliminate duplicates, the Transrelational becomes larger because of the extra space occupied by the *rrt* table consisting of integers, and represented on the analytical model as + (2*IntegerValue), but also to the fact that the compressed values on the fvt table also need to add 2 additional integers to its own size. These integers are the row ranges indicators described in 5.3, and illustrated in Figure 31. This extra size is represented on the analytical model as +(2*IntegerSize*#UniqValues).

7.3.3.2 Data Density

The Binary-Relational model achieved the highest Data Density. It is important to highlight that this higher Data Density is achieved by the way the model stores data in contrast with other approaches followed by commercial RDBMS which attempts to increase its data density by incorporating compression algorithms into its RDBMS [Poss, 2003], but such a data density increase is achieved by technology over the n-ary-Relational data model.

The current thesis considers that increasing data density at the data model level is a better approach than doing it by technology. Technology implementations can fail but a data model will never fail as it is made by definition.

The Binary-Relational model stores one value and relates as many rows as required. An illustration of how this is achieved is presented in Figure 48, l in which two queries are applied to an n-ary-Relational, and the first query shows 6,001,215 rows for the model but the second query shows that only 2,466 distinct values exist. The second query consists of the execution of the UNIX command *word count* (*wc*) to count the number of lines (-l) contained on the file which corresponds to the Binary-Relation "L-COMMITDATE"; it reports only 2,466 lines which corresponds to a number of different values. Then the theory beneath the Binary-Relational model, as stated in section 3.4, has been proved as it does not store duplicates at the column level. Data Density is independent of the Scale Factor used on the Data Warehouse. In Figure 49, this aspect can be appreciated as the patterns are similar no matter the scale factor.



```
SQL> select count      (L_COMMITDATE) from LINEITEM;
COUNT(L_COMMITDATE)
-------------------
        6001215
SQL> select count(distinct L_COMMITDATE) from LINEITEM;
COUNT(DISTINCTL_COMMITDATE)
---------------------------
                2466
```

```
$ strings 34.theap >
L_COMMITDATE.lis
$ wc -l L_COMMITDATE.lis
2466 L_COMMITDATE.lis
$
```

Figure 48 n-ary-Relational Storage vs. Binary-Relational Storage

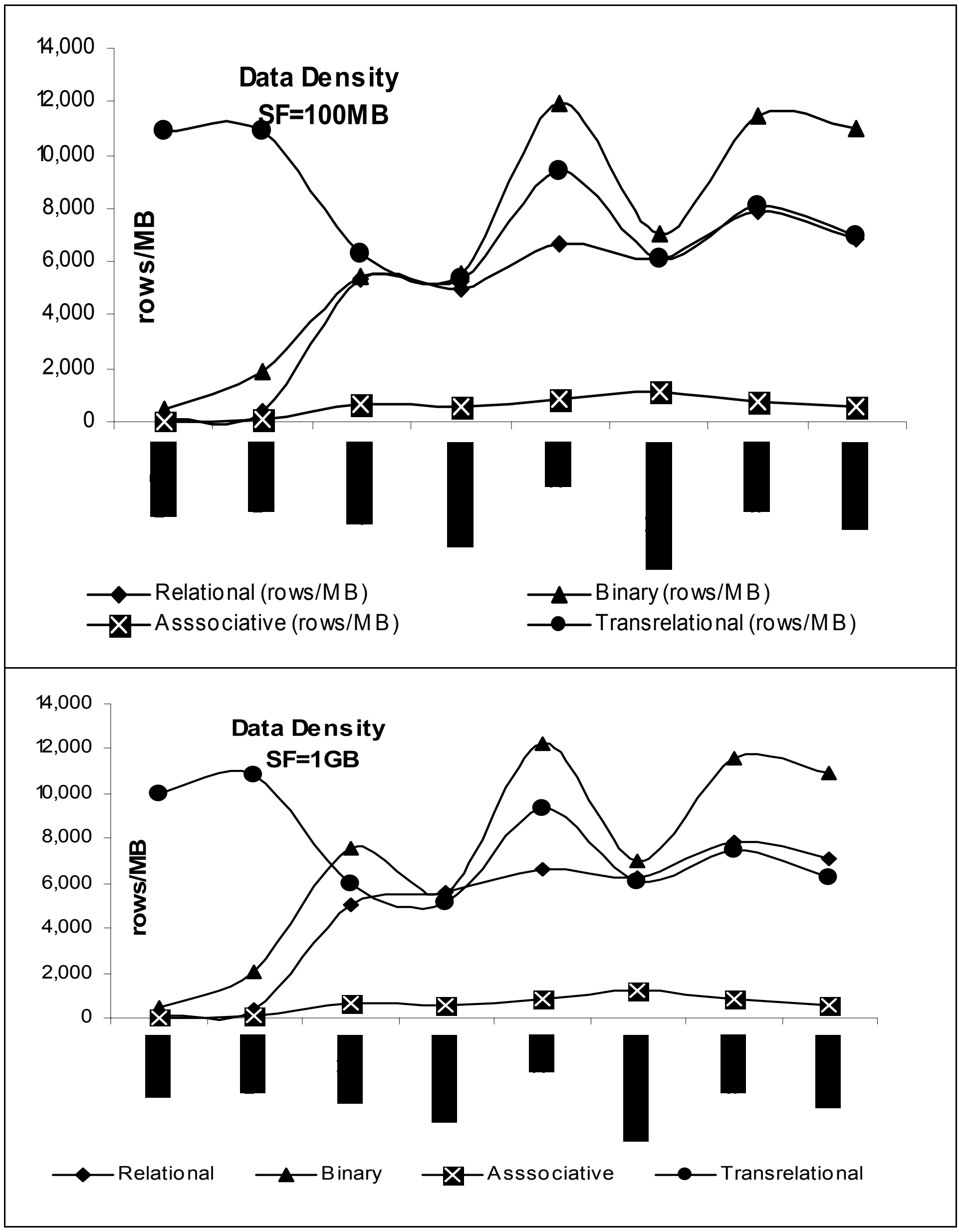


Figure 49 Data Models Data Density

7.3.3.3 Data Sparsity

The research objective 2 relates to the data sparsity which is also related to the data density, as sparse data consumes disk space. When measuring data density in the experiments, it was found that the TPC-H synthetic data set has no data sparsity as its populating programs filled every single field with a value; hence there are no sparse data which allow the evaluation of the behaviour of the alternative data models regarding the management of sparse data.

Real life data sets which contain sparse data have been measured and reported in Chapter 8, which allow the evaluation of how the Binary-Relational model has better handling of sparsity.

As reported in Gonzalez [2005b], Data Sparsity is better handled by the Binary-Relational model than the n-ary Relational model; in fact data sparsity is eliminated by the design of the vertical decomposition data storage [Copeland, 1985] implemented in MonetDB [2004]. If there are no data for a particular column, then nothing is stored. Figure 50 is a graphical representation of the data sparsity management in the Binary-Relational model.

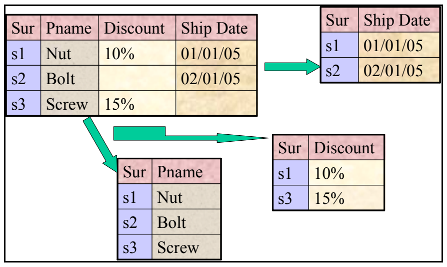


Figure 50 Binary-Relational Model's Data Sparsity Management

7.3.4 Exploitation Phase

The exploitation phase is the phase that finally exposes the Data Warehouse to the end user and can highlight the deficiencies of the Data Warehouse considering the logical data model (star or snowflake) and the data model underlying the DBMS where data are stored.

7.3.4.1 Query Execution times in pristine mode

When running the Power test of the TPC-H, the Binary-Relational model has a consistent query performance, beating the n-ary-Relational model in all queries. This can be appreciated in Figure 51 where two data series are shown in the graph, one with “peaks” and “valleys” corresponding to the n-ary-Relational instantiation and the other, which is almost over the *x* axis, corresponding to the Binary-Relational instantiation. This behaviour is similar, regardless of the scale factor.

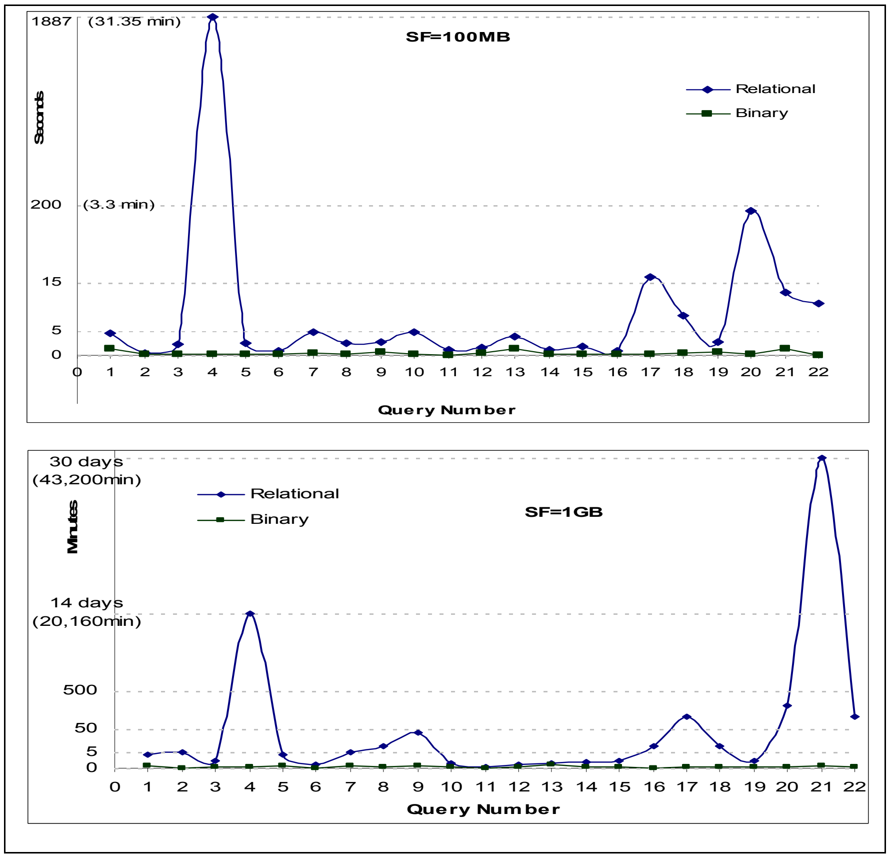


Figure 51 Query times Binary vs. Relational

When executing the 22 queries defined in TPC-H for a scale factor of 1GB, the time difference between the Binary-Relational and the n-ary-Relational is high. This time difference is so high that these orders of magnitude in query response time could be appealing for many organizations dealing with long query execution times, and according to these results, the Binary-Relational model can challenge the established n-ary-Relational model within the context of Data Warehousing.

There are many reasons behind these improvements in the query performance of the Binary-Relational model:

1. Only the columns involved in the query are accessed. By contrast, any horizontal storage based model (n-ary-Relational included) needs to access all columns contained in the record.
2. Duplicates are eliminated at column level in the Binary-Relational model, which reduces the disk space required to store data, as shown in Figure 48. For

example, a date column can have millions of values for a year (in the example 6 million values). These millions of values hardly fit into real memory, but only 365 different values exist in a year, which can fit in real memory. Consequently, many operations can be executed at RAM speed instead of the slower disk speed involved when millions of values stored on disk need to be moved from disk to memory.

3. By organizing data vertically (in columns) and by storing unique values, it is the equivalent of having an index; hence storing tables in a Binary-Relational model based DBMS is approximately speaking the equivalent of having all tables indexed by all columns, which is almost impossible to have in a n-ary-Relational based RDBMS. Since a Binary-Relational based Data warehouse has indexes on all columns, it is able answer any query quickly. Hence this data model is ideal for *ad hoc* queries which are a common requirement in Data warehousing environments.
4. If people try to emulate this behaviour with an n-ary-Relational based DBMS, it is necessary to create indexes over each column. Such an idea is not feasible as too much space is required, and a lot of time is required to build the indexes. During insertions and deletions, the index structures need to be maintained when slowing down these operations. Even worse during query resolution, the DBMS query optimizer builds the query plans based on the indexes available for the operation [Date, 2005]. However only one index is used to solve the query, and if many indexes are available the Query optimizer can make the decision not to use any index and perform a full table scan. Hence having too many indexes on an n-ary-Relational based DBMS can cause problems instead of benefits.
5. In section 6.4.7, the measurement of metrics 11 to 16 was presented. These metrics considered technology specific metrics to improve query performance by the creation of indexes, and computing statistics to improve query performance on the n-ary-Relational DBMS; but even then, none of these scenarios achieved better times that those achieved by the Binary-Relational model instantiation (see also Figure 54 in section 7.3.6).

6. Another reason for queries running faster in the Binary-Relational case is that by having all data organized in columns, the DBMS can start as many processing threads as columns are involved in a query, and then solve the query in parallel mode, similar to the way it loads data (see Figure 44).

7.3.4.2 Temporal Disk Space

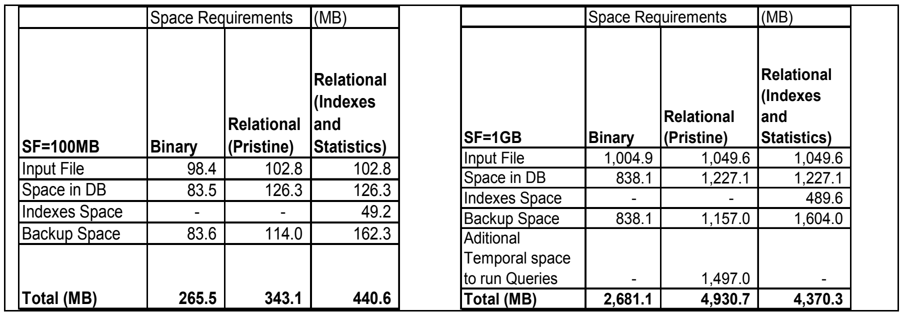
It is also important to analyse the total disk size required, not only by the data itself once loaded into the DBMS, but also considering the space required by the input flat files. These measurements were made in metrics 3 and 5. However, another disk space requirement became apparent during querying the data: it is the temporary space required by the n-ary-Relational DBMS.

As explained before, all columns of the rows involved in a query need to be accessed during the query resolution, and during the query execution a lot of processing needs to be done in temporary areas of the database, for example joins and sorts.

During the query execution measurements for the n-ary-Relational, it was necessary to increase the size of the temporary space up to 1,497MB. The size of the temporal space is considerable, as it is 142% of the size of the input data.

In contrast, the Binary-Relational DBMS did not require extending the temporary area. MonetDB, which was the Binary-Relational instantiation used for the experiments, uses the temporary space of the operating system (*/tmp*).

The total disk space requirements are presented in Figure 52. As can be seen, the n-ary-Relational (in Pristine mode for SF=1GB) needs 2,249.6 MB more space than the binary-Relational (nearly double) and this additional disk space requires extra investment by organizations.



	Space Requirements	(MB)	
SF=100MB	**Binary**	**Relational (Pristine)**	**Relational (Indexes and Statistics)**
Input File	98.4	102.8	102.8
Space in DB	83.5	126.3	126.3
Indexes Space	-	-	49.2
Backup Space	83.6	114.0	162.3
Total (MB)	**265.5**	**343.1**	**440.6**

	Space Requirements	(MB)	
SF=1GB	**Binary**	**Relational (Pristine)**	**Relational (Indexes and Statistics)**
Input File	1,004.9	1,049.6	1,049.6
Space in DB	838.1	1,227.1	1,227.1
Indexes Space	-	-	489.6
Backup Space	838.1	1,157.0	1,604.0
Aditional Temporal space to run Queries	-	1,497.0	-
Total (MB)	**2,681.1**	**4,930.7**	**4,370.3**

Figure 52 Total Disk Space Requirements

7.3.5 Maintenance Phase Data Analysis

Backup and Restore are other tasks that are frequently out of the main research focus in the Data Warehouse area. Metrics 8, 9 and 10 have been defined to consider these aspects in the evaluation. The results are presented in Figure 53 for both Scale Factors, considering Relational and Binary-Relational models. Binary-relational has better backup and restore times than Relational with around 80% less time, regardless of the scale factor.

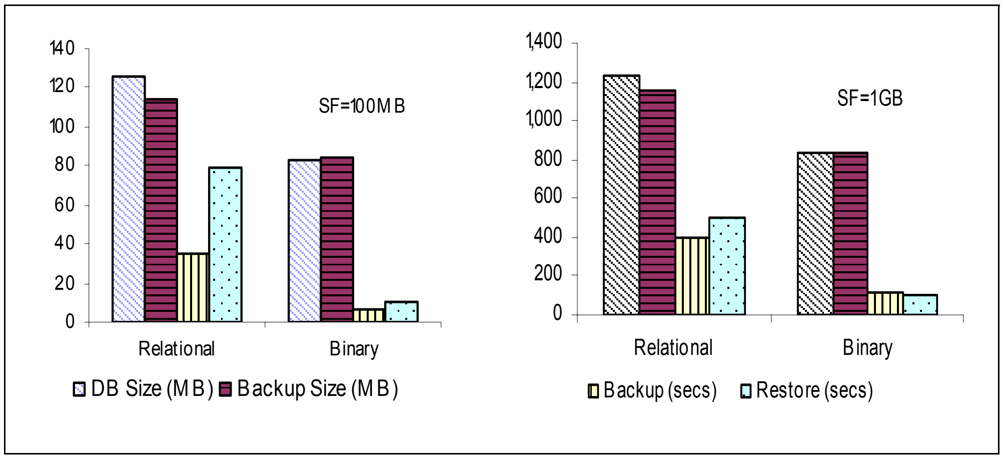


Figure 53 Backup and Restore times

From Table 31 (in Chapter 6), the Binary-Relational backup occupies the same size as the database. This is because it is a dense model and its implementation fills at 100% all data pages, while the relational does not. Typically, RDBMS fill data pages at 80% [Oracle, 2005] and for this reason; backups for relational DBMS are smaller than the actual database size.

Based on these results, research objective 6 is satisfied with regard to solving the maintenance challenges of current Data Warehouse environments.

7.3.6 Technology Specific Metrics to Improve Query Performance

In section 6.4.7, the results of the technology specific metrics were presented, and it has been found that times to create indexes and to compute statistics are high and they can represent more than 100% of the time required to load data. Hence those times are not trivial when using n-ary-Relational based products.

Regarding query times, three scenarios were run in order to measure how these optimisation techniques can improve the n-ary-Relational instantiation. In Chapter 6 detailed data of each query run are presented in Figure 41. By contrast, Figure 54 shows the Total query work load considering the pristine mode of Relational plus the three scenarios described previously and the Binary-Relational times. From here it can be appreciated that Binary-Relational is faster than optimised n-ary-Relational environments.

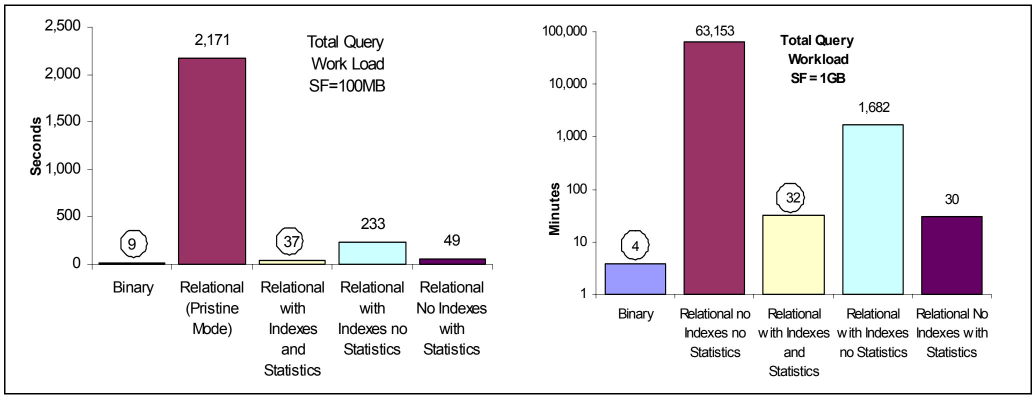


Figure 54 Total Query Workload for Different Relational Optimised Environments

The optimization time (Index + Statistics) is not trivial compared with the required loading time (refer to Figure 40 in Chapter 6). This optimising time is not required by the Binary-Relational model. Apart from that, indexes required extra storage space, and Table 33 and Table 34 show the disk space required by indexes. They can represent between 12% and 100% of the data size. If many indexes are required to optimise queries, the required space can occupy more than the space required by the data themselves.

In both scale factors, huge improvements were achieved for Relational technology over the pristine Relational case, with 5,887% improvement for SF=100MB and 195,222% improvement for SF=1GB; but in both Scale Factors, the better relational times are worse than the Binary-Relational times. For SF=100 MB, Binary-relational is 388% faster than relational technology and for SF=1GB, Binary-relational is 848% faster than relational technology. It should be noted that Binary-relational times are achieved without any further optimisation.

These improvements in query response times are beneficial to those organisations suffering slow query response time.

7.3.6.1 Reductions to the Batch Maintenance Window

Organizations are dealing with long batch maintenance windows in order to maintain their Data Warehouse environment to satisfy their business needs. According to the results obtained in many aspects of the experiments executed, the Binary-Relational model can introduce improvements to this maintenance time. Maintenance is undertaken typically during a night window, but sometimes these nightly processes are still running during the day. Thus if Binary-Relational can reduce this time, organizations can use this extra time to do other kinds of computer processing that bring more business benefits than time spent maintaining indexes, statistics or pre-computing results to try to remedy the deficiencies underlying their RDBMS data model (n-ary-Relational) and the way it works.

In Table 39, an abstract of the tasks and times required during a Batch maintenance window is presented for both Relational and Binary-Relational Data Warehouses.

From Table 39, considering the whole maintenance window, the Binary-Relational model requires around 60% less time than the n-ary-Relational model. This is another aspect in which the Binary-Relational model can challenge the n-ary-Relational model within the context of Data warehousing.

Table 39 Batch Maintenance window required

Data in minutes	SF=100MB		SF=1GB	
	Relational	Binary-Relational	Relational	Binary-Relational
Extraction and Transformation time	2.5	2.6	27.6	29.5
Loading time	2.4	0.5	19.9	5.1
Index Creation time	0.5	0	8.5	0
Statistics Computation	0.8	0	15.6	0
Backup time	0.5	0.1	6.6	1.8
Restore time	1.3	0.1	8.4	1.7
Total Time	8.0	3.3	86.6	38.1
% Saving in batch maintenance window		**59%**		**56%**

7.4 Choosing the Alternative Data Model for an Alternative Reference Architectural Configuration for Data Warehousing (ADW)

After benchmarking the data models, it is possible to choose the data model best suited for Data warehouse environments, and then to propose an Alternative Reference Architectural Configuration for Data Warehousing (ADW).

It is necessary to choose the model by an objective selection. An abstract of the 16 metrics defined in the extended version of the TPC-H (section 6.3.3) is presented in Table 40 and in Table 41 for the two different Scale Factors used and considering n-ary-Relational and Binary-Relational models. These tables are presented as scored cards for easier reading. In these scored cards a checkmark (√) has been assigned to the data model which is better in each metric.

Table 40 Scored Card for SF=100MB

Metric #	Extended TPC-H metrics (SF=100MB)	Unit of measure	Relational	Score	Binary-Relational	Score
1 and 2	Extraction from transactional systems and Transformation times	Minutes	2.5	√	2.6	
3	Input file sizes measurement	MegaBytes	102.81		98.35	√
4	Load data times in to the Data Warehouse	Minutes	2.4		0.5	√
5	Size of the Database tables after load (Data Base size)	MegaBytes	126.31		83.52	√
6	Data Density Measurement	Rows/Megabyte	6,861		10,376	√
7	Query execution times (Pristine mode)	Minutes	36.19		0.16	√
8	Data Warehouse Backup time	Minutes	0.5		0.1	√
9	Backup size	MegaBytes	114		83.64	√
10	Data Warehouse restore time	Minutes	1.32		0.17	√
11	Index creation times	Minutes	0.5		0	√
12	Index sizes	MegaBytes	49.2		0	√
13	Query times (with indexes)	Minutes	3.89		0.16	√
14	Statistics computation times	Minutes	0.8		0	√
15	Query times (with indexes & statistics)	Minutes	0.61		0.16	√
16	Query times (with statistics without indexes)	Minutes	0.81		0.16	√

Table 41 Scored Card for SF=1GB

Metric #	Extended TPC-H metrics (SF=1GB)	Unit of measure	Relational	Score	Binary-Relational	Score
1 and 2	Extraction from transactional systems and Transformation times	Minutes	27.6	√	29.5	
3	Input file sizes measurement	MegaBytes	1,049.58		1,004.95	√
4	Load data times in to the Data Warehouse	Minutes	19.9		5.1	√
5	Size of the Database tables after load (Data Base size)	MegaBytes	1,227.10		838.1	√
6	Data Density Measurement	Rows/Megabyte	7,058		10,334	√
7	Query execution times (Pristine mode)	Minutes	63,152.69		3.81	√
8	Data Warehouse Backup time	Minutes	6.68		1.86	√
9	Backup size	MegaBytes	1,157		838.1	√
10	Data Warehouse restore time	Minutes	8.41		1.72	√
11	Index creation times	Minutes	8.52		0	√
12	Index sizes	MegaBytes	484.6		0	√
13	Query times (with indexes)	Minutes	1,682.31		3.81	√
14	Statistics computation times	Minutes	15.63		0	√
15	Query times (with indexes & statistics)	Minutes	32.35		3.81	√
16	Query times (with statistics without indexes)	Minutes	29.57		3.81	√

As can be appreciated from both score cards, the Binary-Relational model achieves better measurements in all metrics but metrics 1 and 2 which are related to the response time of the transactional source systems, and in this case any transformation was required as the source system was already within a Relational DBMS and had the same data structure of the TPC-H database.

Based on these metrics results, it has been demonstrated that an alternative data model can be more appropriate for Data Warehouse environments. In this case, it is the Binary-Relational Model.

In the next chapter, an Alternative Reference Architectural Configuration for Data Warehousing (ADW) will be proposed. It uses the Binary-Relational model as its base. Also, this alternative reference architectural configuration has been tested in industry under real life scenarios and the results obtained by ADW are also presented in Chapter 8.

7.5 Conclusions

In this chapter the results of the experiments have been analysed and it has been found that the Extraction and Transformation times are independent of the target data model; they depend more on the speed of the source systems, but regarding Loading times, they are improved when the target model is the Binary-Relational model, as it starts one loading thread per each column within the row, for example if a row has five columns, it will start five loading threads. It has been suggested that an ELT approach be adopted to load the Data warehouse, as usually the target machine has more computer power and the Binary-Relational DBMS will process the Transformation task faster.

Regarding the Storing phase, the Binary-Relational model achieved the highest data density contributing to a smaller Data Warehouse, which challenges the presumption that data will grow once they are loaded into a DBMS.

Also because of its vertical approach to managing data, the Binary-Relational DBMS has achieved better query times than the other models, and with this, organisations can improve their business analysis by having a faster model to respond to the queries.

The reduction on the Data Warehouse size also benefits the organisations by reducing the maintenance task, regarding mainly the Backup and Restore tasks. Also because of its vertical approach and the elimination of duplicates at column level (this is, roughly speaking, the equivalent of having indexes on every single column of all tables) makes the Binary-Relational model ideal for an *ad hoc* query and reporting environment.

In the global evaluation of the n-ary-Relational and the Binary-Relational models, this last one has obtained better results in almost all the aspects defined on the extended version of the TPC-H that was created during the research for this thesis.

As one of the alternative data models, the Binary-Relational has satisfied the initial hypothesis of increasing data density and reducing data sparsity. The use of such a model will be considered within an Alternative Reference Architectural Configuration for Data Warehousing, which will be described in Chapter 8.

CHAPTER 8
The ADW Reference Architectural Configuration

8.1 Introduction

As exposed in section 1.4.1, current Relational Data Warehouses are experiencing the following problems:

1. Data Warehouses grow at an exponential rate [Datta, 2004].
2. The Database explosion phenomenon [Pendse, 2005] is difficult to control or eliminate.
3. Poor management of data sparsity [Gonzalez, 2005b].
4. Low Data Density [Gonzalez, 2005b].
5. Huge amounts of disk storage are required [Zukowski, 2005].
6. The cost of storage and its maintenance are not negligible [Datta, 2004] and the storage itself could be up to 80% of the entire system cost [Zukowski, 2005].
7. One of the main complaints from uses is the high query processing times. According to Pendse [2003b], 17% of all sites using OLAP tools complain about the query response time, with the worst case reaching 42% of the sites that use Oracle Discoverer.
8. Long periods of time to Extract, Transform and Load (ETL) data [Gonzalez, 2005b].
9. Big batch processing windows to backup and restore the environment [Gonzalez, 2005a].
10. High complexity of the Database Administration tasks, including Index creation and its maintenance, and long times required to compute statistics.

These problems are solved by proposing an Alternative Reference Architectural Configuration for Data Warehousing (ADW) which is part of the current research work of this thesis.

8.2 ADW Purpose and Problems Solved

The objective of proposing an Alternative Reference Architectural Configuration for Data Warehousing (ADW) is to resolve the ten points listed above. After benchmarking some alternative data models and their behaviour in Data warehouse environments, it was found that the Binary-Relational model is best suited for its use in Data warehouse environments (refer to Chapter 6 and Chapter 7). Based on these findings, an alternative reference architectural configuration for Data Warehousing which includes a Binary-Relational repository is proposed.

In this section, the revision of how the ADW reference architectural configuration addresses the ten points listed in section 8.1 is made.

1. *Data Warehouses are growing at an exponential rate*. This problem is remediate in ADW by using a Binary-Relational repository, as it has been demonstrated in section 6.4.3.1 where the Data warehouse size was measured. Savings of 32% are achieved when compared with an n-ary-Relational Data warehouse. These measurements considered base tables only, but usually in n-ary-Relational Data warehouses, consolidated tables or materialised views are created as well. These extra tables are reduced or even completely eliminated in an ADW. Because of its fast query response time, it is not necessary to build these summary tables.
2. *The Database explosion phenomenon is difficult to control or eliminate*. In the case of n-ary-Relational architectures, when computing the cross product of all levels of the dimensions against each other -the cube computation- is one of the main contributors to the database explosion. In the case of an ADW and its Binary-Relational architecture, such cube computation is not necessary because of the speed to answer queries.
3. *Poor management of data sparsity*. n-ary-Relational Data warehouses have poor management of data sparsity because they are based on the n-ary Relational model which has always suffered in the handling of missing information [Codd, 1990]. However, and even worse, the relational implementations based on SQL have poor Nulls management that is directly related to data sparsity. In contrast, the Binary-Relational has a better Data sparsity management by the model definition; see Figure 50, where sparsity is eliminated at the model level.

4. *Low Data Density*. As demonstrated by the measurements made in section 6.4.3.2, the Binary-Relational model has better data density (10,000 rows/MB) than the n-ary-Relational model (7,000 rows/MB). Hence the use of a Binary-Relational DBMS will improve the data density of the Data warehouses, which is precisely the kind of DBMS to be used in the ADW reference architectural configuration.
5. *Huge amounts of disk storage are required*. When using the Binary-Relational model, savings of 65% of total disk space are achieved when compared against n-ary-Relational. This has been measured and shown on Figure 52 (Chapter 7).
6. *The cost of storage and its maintenance is not negligible and the storage itself could be up to 80% of the entire system cost*. As stated in the previous point savings of 65% in disk space can be achieved by having ADW implemented. These savings are directly reflected in the total cost of the disk and its maintenance. Some other savings are listed below.
    a. 65% less disk space to be acquired.
    b. Electricity savings achieved by having fewer disks to be powered.
    c. Reduction in energy costs by reducing the cooling needed by the computer centre having fewer disks generating heat, and even savings when buying a smaller cooler machine.
    d. Savings in computer room floor space by having to accommodate fewer disk arrays.
    e. Savings in secondary storage (tapes) in the same proportion by having fewer disks to be backed up.
    f. By having fewer tapes to store, reductions in the size of the room to keep the historical tapes can also be considered.
    g. By having 65% less disks, the payment of the corresponding maintenance fee to the hardware vendors is also reduced.
7. *One of the main complaints from users is the high query processing times*. This aspect can be dramatically improved when implementing ADW, because the Binary-Relational model beneath ADW has achieved query performance improvements in many orders of magnitude - 3.8 minutes versus the 43.8 days

of the n-ary-Relational model (refer to sections 6.4.5.1 and 7.3.4.1 where the actual measurements were presented).

8. *Long periods of time to Extract, Transform and Load (ETL) data*. Based on the measurements made in sections 6.4.1 and 6.4.2 and the analysis made on sections 7.3.1 and 7.3.2, ADW reduces the ETL time required to Load the Data warehouse.
9. *Big batch processing windows to backup and restore the environment*. This aspect can also benefit from implementing ADW based on the experimental results made in sections 6.4.6.1 and 6.4.6.3, where savings of between 70% and 85% were achieved in these aspects of the Data warehouse maintenance.
10. *High complexity of the Database Administration tasks, including Index creation and its maintenance and long times required to compute statistics*. ADW is able to reduce the Database administration tasks because it does not need to create extra indexes, or to compute statistics. The reasons for these tasks not being required by the Binary-Relational model were analysed in section 7.3.6. When implementing ADW, organisations can forget about creating and maintaining extra indexes and computing statistics, hence reducing the database administration tasks and even reducing the maintenance time required by the whole environment.

8.3 Keeping Compatibility with the Installed Base

Keeping compatibility with the installed base is research objective 5. This aspect is important because organisations have invested vast sums of money in their current technology. Such technology is compatible mainly with the n-ary-Relational model approach, or more precisely with the SQL language.

The majority of the existing tools use SQL, no matter which part of the Full Data Warehouse Cycle they are oriented to (refer to section 6.3.1). ADW needs to consider such architectural pieces and the way to integrate them, and not how to replace them.

Therefore it is important for any Binary-Relational DBMS to have a SQL interface to enable existing commercial tools to be integrated. This could be named as backward compatibility, but future Binary-Relational based DBMS could also offer their own

Binary-Relational language to enable the evolution of the exploitation or loading phases.

8.4 The Reference Architectural Configuration Description

An Alternative Reference Architectural Configuration for Data Warehousing (ADW) has been defined. It is depicted in Figure 55, and a description of the reference architectural configuration is given here.

The ADW reference architectural configuration considers an ELT stage, a storage stage (or the Data Warehouse) plus an exploitation stage.

Organisations' information has its origin on the working transactional systems, some times called 'operational systems' or 'OLTP environments'. From here data must be Extracted, Transformed and Loaded according to the specifications of the logical Data Warehouse model (star or snowflake) and with the requirements of the data model of the repository, which in the case of ADW is a Binary-Relational DBMS. The decision to use a Binary-Relational DBMS has been based on the results obtained during the experiments carried out in this research (see Chapter 6 and Chapter 7), and where the Binary-Relational model has achieved the best results to manage Data Warehouses.

It is important to stress that the ADW reference architectural configuration can support both the ETL or ELT approaches to acquiring data and feeding them into the Data Warehouse, but in the case of Data Warehouses it is better to do bulk loads and perform bulk transformations; hence the ELT approach is preferred.

Once data has been loaded into the Binary-Relational repository, it is necessary to enable such DBMS to be accessed by existing reporting and analysis tools based mainly on the SQL language. Therefore ADW considers a Query interface Module which will manage the translation of SQL statements to the Binary-Relational language supported by the DBMS; the MonetDB uses a language called MIL, but other DBMSs can have their own language.

ADW is not limited to SQL language tools; if in the future vendors develop exploitation tools which talk directly to the Binary-Relational language, and then the translation between SQL and this language will be no longer required.

In the exploitation phase, different kinds of tools are supported, e.g. Report Writers, CRM, Molap and Rolap tools or Data mining tools, depending on the kind and sophistication of the analyses required by the end user.

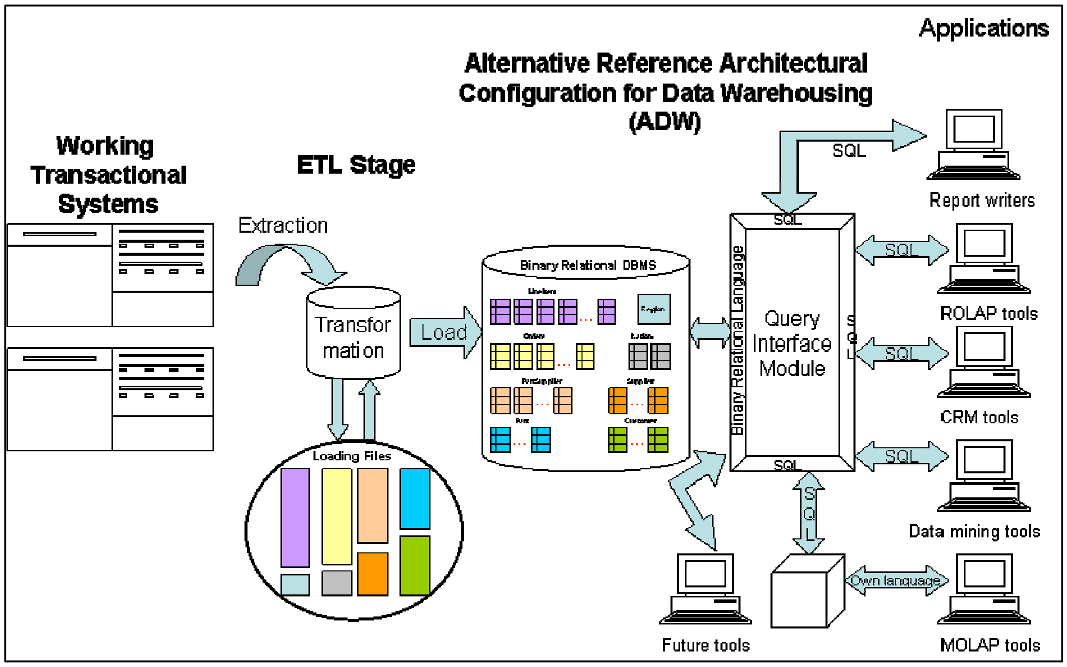


Figure 55 The ADW Reference Architectural Configuration for Data Warehousing

8.4.1 ADW Logical Modelling Support

As mentioned in section 2.3.1, logical modelling design is an important part of a Data Warehouse. ADW is an open reference architectural configuration which supports different types of logical data warehouse models, including Star and Snowflake modelling techniques, or even traditional Entity-Relationship models, in which it still offers the advantages of using Binary-Relational-like disk space reduction and Query performance improvements.

8.4.2 ADW Benefits to Query Performance

As ADW is based on a Binary-Relational repository, it will provide the query performance improvements achieved by this alternative data model, as highlighted in sections 7.3.4 and 7.3.6.

ADW will answer queries faster if the Binary-Relational repository is accessed by tools which generate direct queries to the DBMS, but special care must be taken when using cube-based tools. The cube building must avoid the kind of queries that select all rows and columns in order to make the cross product operation to build the cube. The operation of retrieving all columns of a row and then retrieving all rows of the table will force the Binary-Relational DBMS to rebuild all records and then compute the cross product, which will be an extremely time-consuming task; consequently the benefit of fast query performance achieved by the Binary-Relational based DBMS would be diminished by the cube construction.

8.4.3 ADW Support for the existing query performance approaches

ADW is an architectonical configuration which can support any of the existing approaches to improving query performance. These existing approaches were described in section 2.4.

In section 2.4.1, the use of summarized tables was described. Even when it is not recommended that summarized tables be used, ADW can support them, as these tables are other tables which do not have any difference from the point of view of the DBMS; however, even the Binary-Relational model will help to build these summarized tables as it will compute the summary information in a faster fashion. Owing to the query performance of the binary-relational DBMS, the use of summarized tables will be reduced or even eliminated in ADW. This aspect will contribute to the disk and time savings of the whole reference architectural configuration.

In section 2.4.2, the materialized views approach to improve query performance was described. ADW can also support this approach and, as stated, the materialized views are similar to the summarized tables. Hence Binary-Relational DBMSs can integrate this technology, but again, owing to the query performance of the Binary-Relational model, materialized views should be not necessary.

The other existing approach to improve query performance is to do Approximate Queries (section 2.4.3). This approach can be used over ADW, but even though the Binary-Relational performance is superior, queries can be answered completely instead of computing just an approximation.

Finally, the use of cubes to improve query performance is not excluded from ADW, as can be appreciated from Figure 55. If the decision is made to continue using cubes on ADW, then the benefit that ADW will provide is that the cube will be build faster as the Binary-Relational DBMS will answer faster.

ADW will be more beneficial for ROLAP tools as they send queries directly to the DBMS. In some situations when ROLAP tools are used, users complain about the slow performance of the system, but usually the slow performance is caused mainly by the RDBMS used and not to the exploitation tool. Therefore in ADW, the DBMS has been improved by using a Binary-Relational model based DBMS, which is faster.

8.4.4 ADW Approach to increase the data density

The research objective 1 is related to the data density increase, and according to the results achieved by the Binary-Relational model, and, as ADW incorporates a Binary-Relational DBMS, then ADW will help to increase Data Density.

Data density has been measured considering only data from base tables, but if a generalisation is made and not only data but information is considered, then the density will be increased in a larger proportion, as ADW will not use summary tables or materialized views or even cube structures, as they are not necessary; then the amount of useful information stored per disk Megabyte is even higher.

The highest data density of ADW is achieved because it stores only different values at column level and then it relates each value to as many records as required.

The work conducted in this thesis differs from the current approaches to increase data density (which basically are data compression techniques and were explained in section 2.5), in the sense that in this research a change in the underlying model is made and models which abandon the classical record storage structure have been considered.

Recent work in commercial DBMS [Poss, 2003] considers the use of compression techniques on the DBMS. In Poss [2003], it compresses data which are clustered in near disk areas. The study of using compression techniques on DBMS have been mentioned in section 2.5.2 and in this chapter some other drawbacks have been found during the industrial testing of ADW. These will be highlighted in section 8.5.2

8.5 ADW tested in Industry

The following sections are based on industrial experience, in which initial tests of the alternative reference architectural configuration for Data Warehousing (ADW) have been made.

After the results were obtained from the laboratory using the synthetic TPC-H data set and where the Binary-Relational Model emerged as the best model for Data Warehousing environments, it was appealing to move forward and validate the conclusions obtained in the laboratory by using ADW within industrial environments.

Industry implementations have been made for six organisations in five different industries: one in government, one in manufacturing, one in retail, one in telecommunications and two in the financial industry. These real life-results are similar to the results obtained in the laboratory at Heriot-Watt University.

The Binary-Relational model instantiation used was Sybase IQ version 12.6, as this is a mature product commercially available in the market, but the results are similar to those achieved by MonetDB, which was the Binary-Relational model instantiation used in the laboratory. These concordant results confirm that the results are achieved by the Binary-Relational model itself and not by specific technology implementations.

Because of confidentiality, the names of the organisations have been removed and in some cases the queries have been renamed to keep the data anonymous.

8.5.1 ADW in a Manufacturing Industry Company

This is a manufacturing company with operations in more than 50 countries. The main problems of this company were the high query response time plus the disk space required to store data. They were using a traditional n-ary-Relational DBMS: Microsoft SQL Server.

Manufacturing Industry Company's Test Environment

Sybase's Power Designer tool has been used to do a reverse engineering, plus the forward engineering in order to generate the 1,470 table scripts required by the Binary-Relational model instantiation (Sybase IQ).

All tests were made with dedicated servers and no other application was running during the tests. An important remark is that the hardware capacities of the production environment were double those of the capacities of the test environment (see Table 42).

Table 42 Manufacturing Company Hardware Configuration

	Production Environment	Test environment
DBMS	n-ary-Relational instantiation: Microsoft SQL Server	Binary-Relational Instantiation: Sybase IQ
Operating System	Windows	Windows
CPUs	8 CPUs	4 CPUs
RAM	8GB	4GB

A smaller machine was used because this manufacturing company does not have another machine with the same hardware power as the one they are using on production. Based on the results obtained on the laboratory test (Chapter 6), this company has undertaken the test with half of the hardware, and the results were superior.

Manufacturing Industry Company's ADW Configuration

Figure 56 shows the ADW configuration used for the Manufacturing Industry Company, and as can be seen, not all the elements presented in Figure 55 are used in this configuration. The names of the commercial products used during the materialization of each element have been included; as ADW is a reference architectural configuration, it can use different products in a particular configuration.

In this case, an ETL approach has been followed as it was not the intention to change all the ETL programs. ETL phases were materialized by Cognos' Decision Stream Product.

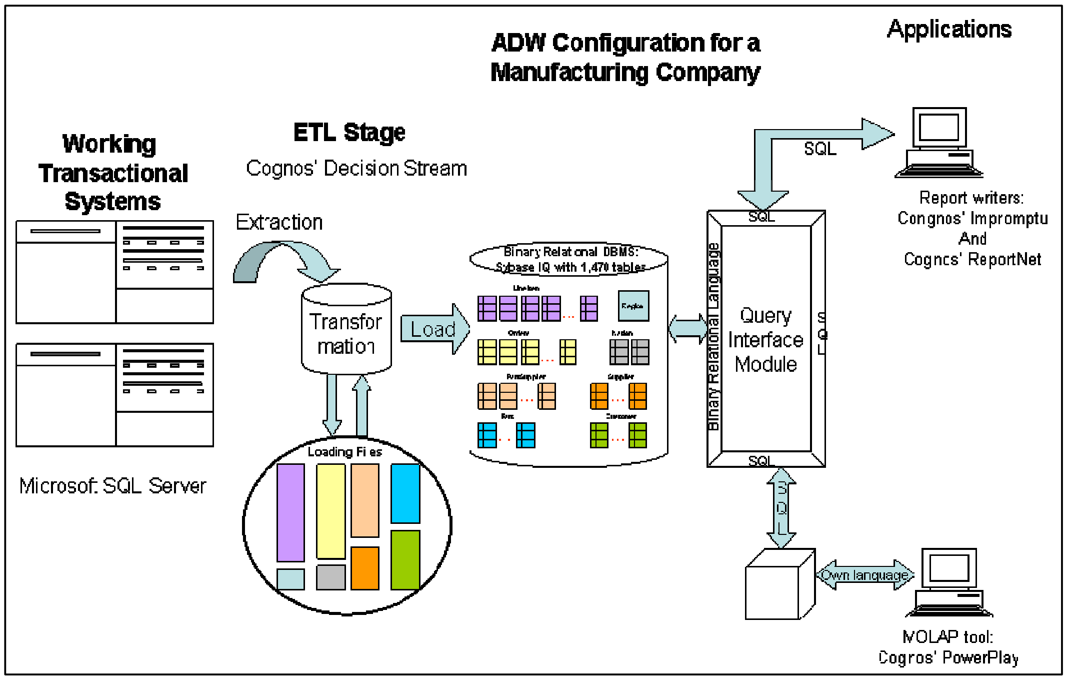

ADW Configuration for a Manufacturing Company
Applications
Working Transactional Systems
ETL Stage
Cognos' Decision Stream
Extraction
Transformation
Load
Loading Files
Binary Relational DBMS: Sybase IQ with 1,470 tables
Binary Relational Language
Query Interface Module
SQL
Report writers: Congnos' Impromptu And Cognos' ReportNet
Microsoft SQL Server
Own language
MOLAP tool: Cognos' PowerPlay

Figure 56 ADW Configuration for a Manufacturing Company

Manufacturing Industry Company's ETL Processes

The measurement of ETL processes is relevant for any Data Warehouse; the results of the ETL processes for this industrial test are presented in Table 43.

Table 43 Manufacturing Company's ETL results

	Traditional DW Architecture	ADW	**Savings (%)**
Extraction Time (HH:MM:SS)	4:26:48	4:26:48	0%
Loading Time	4:14:37	0:19:49	**92.57%**
Total ETL time	8:41:25	4:46:37	**48.83%**

This confirms the assertion made in section 7.3.1, where it was established that Extraction and Transformation times are constant, but Loading times are positively affected when using a Binary-Relational Model based DBMS in ADW. This is achieved because the Binary-Relational DBMS can start one loading thread process for each Binary-Relation (column) of the table; thus the general behaviour observed is the parallel data load into the table.

The data loaded was a Latin American corporate Data Warehouse which included seven countries; details for each country are given in Table 44. The Loading times for the ADW have been separated just to highlight that the main processing times are the results of the Extraction portion of the process.

Table 44 Load time vs. Extraction time by Country

Country	Traditional DW arch. (total ETL time) (HH:MM:SS)	ADW (Total ETL time) (HH:MM:SS)	ADW (ET times) (HH:MM:SS)	ADW (Load times) (HH:MM:SS)
Costa Rica	0:44:00	0:21:46	0:18:56	0:02:50
Dominica	0:56:04	0:29:07	0:26:17	0:02:50
Nicaragua	0:28:26	0:15:32	0:12:42	0:02:50
Panama	0:45:21	0:24:06	0:21:16	0:02:50
Puerto Rico	0:45:07	0:25:19	0:22:29	0:02:50
Venezuela	2:46:07	1:37:26	1:34:36	0:02:50
Colombia	2:26:20	1:13:22	1:10:32	0:02:50
TOTAL	**8:41:25**	**4:46:37**	4:26:48	0:19:49

Manufacturing Industry Company's Disk Space

The next phase of the process is to measure the Database space required for each architectural configuration. The results are summarized in Table 45.

Table 45 Manufacturing Company Disk Space

Traditional DW	148 GB
ADW	38 GB
Disk space savings	74.33%

This result shows that higher disk savings can be achieved by ADW configuration, but this depends on the nature of the data. The synthetic TPC-H data set achieved 32% disk reduction but with this company's data set the reduction was higher with 74% less disk space than the Traditional DW configuration.

Manufacturing Industry Company's Query Response Time

One of the most important aspects of a Data Warehouse environment is the Query response time. A set of relevant queries for this manufacturing company was run, and the results are presented in Table 46, Table 47, and Table 48. This manufacturing company uses Cognos as its analysis tool. The results for direct SQL report using Cognos' impromptu [Cognos, 2008] are in Table 46.

Table 46 Direct SQL Reports

	Traditional DW	ADW
Execution time	47 seconds	2 seconds
Fact_rows_retrieved	3,365,660	3,365,660
CPUs	8 CPUs	4 CPUs
RAM	8 GB	4 GB

Queries in ADW ran faster than in the Traditional DW. It achieved savings of 96% in query times. After direct queries, other business analyses were run. The first one is called "Invoicing Analysis" and it consists of many individual queries that provide the full invoicing analysis to the business. The results are presented in Table 47 .

Table 47 Invoicing Analysis Query times

Query	Retrieved rows	Traditional DW (mm:ss.mmm)	ADW (mm:ss.mmm)	Faster (%)
Accounts_Payable_Administration	137	00:12.000	00:00.125	1,200,000%
Overprice_by_Cost_Centre	2	00:01.000	00:00.156	50,000%
Billing_By_Cost_Centre	73	00:01.000	00:03.141	-33%
Sales_by_Cost_Centre	66	00:36.000	00:44.484	-82%
Billing_by_Customer	407,142	02:29.000	00:03.297	4,962%
Sales_by_Customer	9,716	00:58.000	02:54.594	-33%
Documents_by_Company	3	00:05.000	00:02.125	250%
Billing_Provision	407,121	07:42.000	01:23.890	557%
Raw_Materials_Overprice	3	00:01.000	00:00.390	25,000%
Sales_Raw_Materials	672,305	08:39.000	01:01:047	850%
Product	8,207	00:02.000	00:00.922	22,222%
Order_type	1,978,557	01:34.000	00:03.969	2,350%
Truck	3,261	00:01.000	00:00.234	50,000%
	TOTAL	**22:21**	**6:13**	**358%**

As can be seen in Table 47, ADW outperformed the Traditional DW in the total time for the invoicing analysis; it ran 358% faster. The Query execution time savings are 72% of the query execution time.

Another full business analysis was run; it is the "Accounts Receivable", the results of which are presented in Table 48. ADW had better response times; its total query workload was 313% faster than the traditional DW. It achieved savings of 67% in query execution time. Another important fact is that the applications did not require any modification; they were just redirected from the Microsoft SQL Server ODBC to the Sybase IQ ODBC, and the reports ran successfully.

Table 48 Accounts Receivable Analysis Query Times

Query	Retrieved rows	Traditional DW (mm:ss.mmm)	ADW (mm:ss.mmm)	Faster (%)
Credit_Administration	137	00:02.000	00:00.110	200,000%
Accounts_receivable_by_cost_ Centre	83	00:48.000	00:00.968	4,800%
Accounts_receivable_by_cost centre_monthly_transactions	487,479	00:56.000	00:14.203	400%
Accounts_receivable_by_client	6,369	00:41.000	00:27.344	152%
Accounts_receivable_monthly_ transactions	11,651	00:19.000	00:21.890	90%
Accounts_receivable_ documents_by_company	5	00:10.000	00:01.813	992%
Total_Accounts_Receivable	50,383	01:02.000	00:11.734	563%
Total_Acc_Rcev_month_trans	380,621	03:10.000	00:59.000	322%
Loaded_Days	53,091	00:04.000	00:03.688	133%
Invoicing_rotation	81,269	00:14.000	00:05.985	233%
Transactions_by_currency	177	00:01.000	00:00.141	100,000%
	TOTAL	**07:27**	**02:26**	**313%**

Manufacturing Industry Company's Conclusions

Even with half of the computer hardware power, ADW has demonstrated their benefits in a manufacturing company. In terms of hardware, it will represent computer cost savings of 50% in CPUs and memory, plus 74% savings in disk space.

From the business point of view, queries ran more than 300% faster in ADW than in the traditional DW configuration. Benefits achieved for the manufacturing company are summarized in Table 49.

Table 49 Benefits for a Manufacturing Industry Company

50% savings in CPUs costs
50% savings in RAM Memory costs
74% savings in Disk storage costs
92% savings in Load time
+300% faster queries response times
The applications did not require any modification in order to use ADW

8.5.2 ADW in a Government Agency

The main problems were long query response time plus the disk space required to store data. They were using a traditional n-ary-Relational DBMS: Oracle.

Government Agency's Test Environment

This institution had a star schema for its Data Warehouse's logical model, hence its use in ADW was transparent; the institution provided the SQL creation scripts for the star schema and it was run on the Binary-Relational DBMS.

The test was made on a dedicated server, for both ADW and the Traditional DW configuration. Each test was run with the other DBMS stopped to avoid interference, as it was made in the experiments in the laboratory at the Heriot-Watt University.

During the experiments at Heriot-Watt University, the TPC-H synthetic data set was used. The synthetic data set has no sparsity as it filled all fields with a specific value (section 7.3.3.3). It was thought that this situation could be almost impossible in real life environments. But in the case of this agency they fill all fields with a value in order to avoid missing tuples during join operations [Codd, 1990], and instead of leaving NULLs, they use ZERO values. Hence this data set has no sparsity, similar to the TPC-H data set.

In this case, it was possible to have a dedicated machine to run the same scenarios with both architectures. The hardware and software configuration is in Table 50.

Table 50 Government Agency's Test Hardware Configuration

	Test Environment
n-ary-Relational DBMS	Oracle
Binary-Relational DBMS	Sybase IQ
Operating System	AIX
Computer and CPUs	IBM pSeries670 with 8 Power CPUs at 1.1 GHz
RAM	16GB

Government Agency's ADW Configuration

Figure 57 represents the ADW configuration for a government agency; in this case an ELT approach has been followed because the logical schema was the same and the agency wants to load directly from Oracle and avoid generating intermediate flat files. Some elements of the full reference architectural configuration have been used. The names of the commercial products used to instantiate the ADW elements have been listed on the same figure.

In this ADW configuration, Microsoft Analysis Services has been used to generate the cube structures required to be used by the Hyperion OLAP tool.

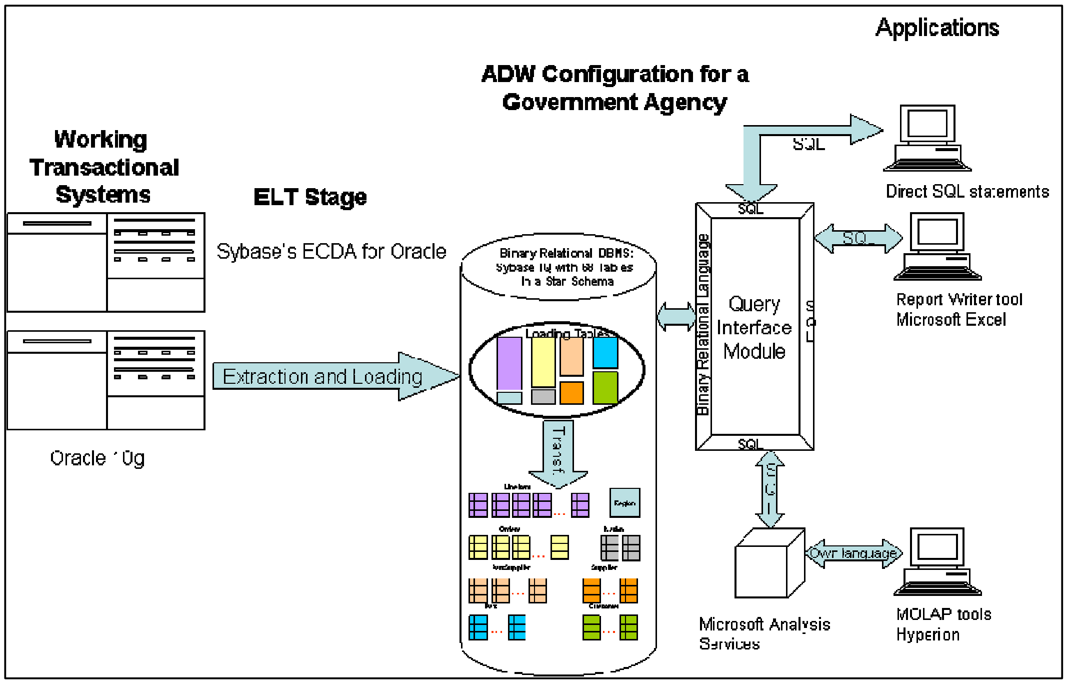

Applications
ADW Configuration for a Government Agency
Working Transactional Systems
ELT Stage
Sybase's ECDA for Oracle
Binary Relational DBMS: Sybase IQ with 88 Tables in a Star Schema
Loading Tables
Extraction and Loading
Transf
Oracle 10g
SQL
Direct SQL statements
Report Writer tool Microsoft Excel
Binary Relational Language
Query Interface Module
Microsoft Analysis Services
Own language
MOLAP tools Hyperion

Figure 57 Government Agency's ADW Configuration

Government Agency's ELT Processes

The results listed in Table 51 include Extraction and Loading times for both Data Warehouse configurations. As the data was already in a similar Data Warehouse logical model (a Star), it was not necessary to make any Transformation, neither for ADW nor for the Traditional DW configuration.

Table 51 Government Agency's Loading results

	Traditional DW	ADW	**Savings (%)**
Loading Time (HH:MM:SS)	27:00:00	12:20:30	**55%**

Savings of 55% were achieved by ADW. This percentage can appear low, but in real terms it represented 15 hours less of processing time for this agency.

Government Agency's Disk Space

Once the data were loaded, the disk space measurement was made. The results for both configurations are presented in Table 52.

Table 52 Government Agency's Disk Space

	Traditional DW	ADW
Database (without compression)	97 GB total (24 GB in data 47 GB in Indexes 26 GB in Temporal)	Not Applicable
Database (with compression by the n-ary RDBMS)	52 GB	12GB

Considering Table 52, the following conclusions can be made:

1. ADW can manage the same amount of information in 12.37% of the uncompressed disk space required by the Traditional DW Configuration (it considers data+ indexes+ temporal space), or in other words it is achieving savings of 87.63%.
2. If considering data space only, savings of 50% are achieved by ADW.
3. The construction of indexes in the Traditional DW configuration can occupy more than the data itself (47GB of indexes vs. 24GB of data). Those indexes are built in order to try to remedy the query performance problem of this agency. Similar patterns exist in many other organizations.
4. Even when using the compression techniques offered by the n-ary-Relational DBMS [Poss, 2003], the Binary-Relational DBMS has achieved 76% savings when compared against the compressed version of the n-ary-Relational DBMS (52 GB vs. 12GB).
5. The n-ary-Relational DBMS used in the Traditional DW Architecture required 5.5 hours to compress data. This time must be added to the 27 hours required to load time, giving then 32.5 processing hours before data is available for querying on the Traditional DW configuration; while in ADW data are ready for querying after 12 hours of processing, achieving then savings of 63% in total processing to have data ready for querying.

Government Agency's Query Response Time

In the case of query time, a set of 128 queries used in production was provided and benchmarked in both architectures. In one of the runs, the queries were limited to a specific geographical entity (filtered) and in the second run this condition was removed so the queries were run at national level. The results obtained are shown in Table 53. These queries are SQL sentences which are sent directly to the DBMS.

Table 53 Government Agency's Queries Set.

	Traditional DW (minutes)	ADW (minutes)	Savings (%)
128 queries with filter	165.47	3.17	98.08%
128 queries without filter	239.07	3.49	98.54%

As can be seen from Table 53, ADW has outperformed the Traditional DW configuration by running the same sets of queries in just 2% of the time required; in other words, instead of running in 3 or 4 hours, the answers were ready in 3 or 4 minutes in both cases. Graphs for both filtered and unfiltered queries sets have been made and they are presented in Figure 58 and Figure 59. As can be appreciated, the behaviour of both graphs is similar to the behaviour presented in Figure 51 in section 7.3.4.1 with the synthetic TPC-H data set where the Binary-Relational model's series is very close to the *x* axis of the graph, while the n-ary-Relational model instantiation series has "valleys" and "peaks", as highlighted in section 7.3.4.1.

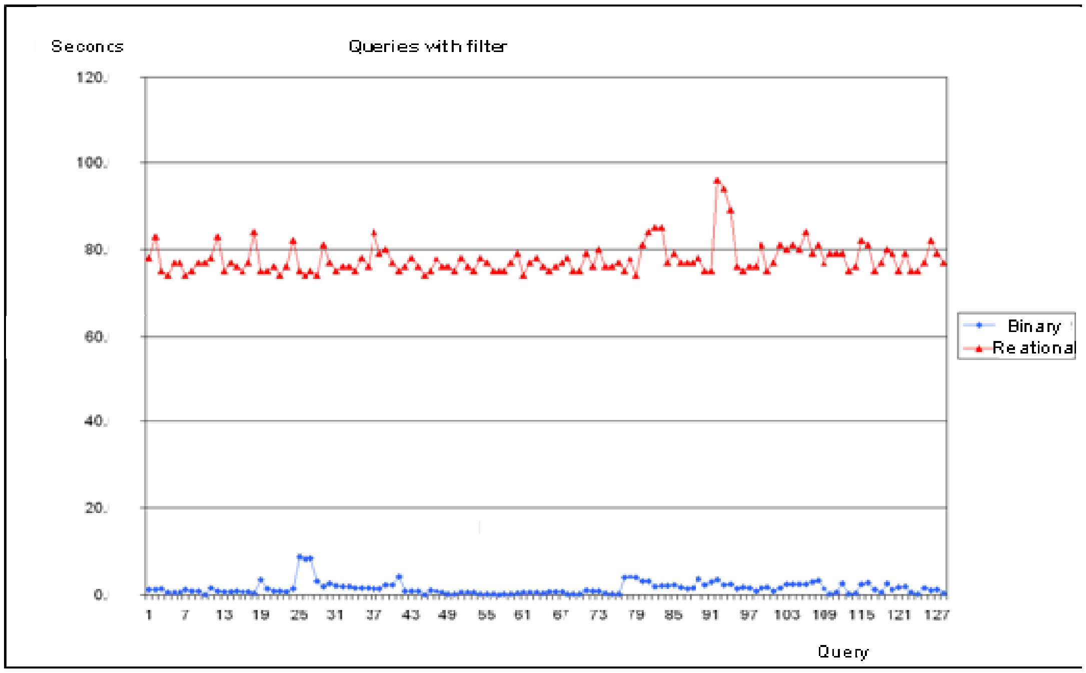


Figure 58 Government Agency's Query Set Run With Filter

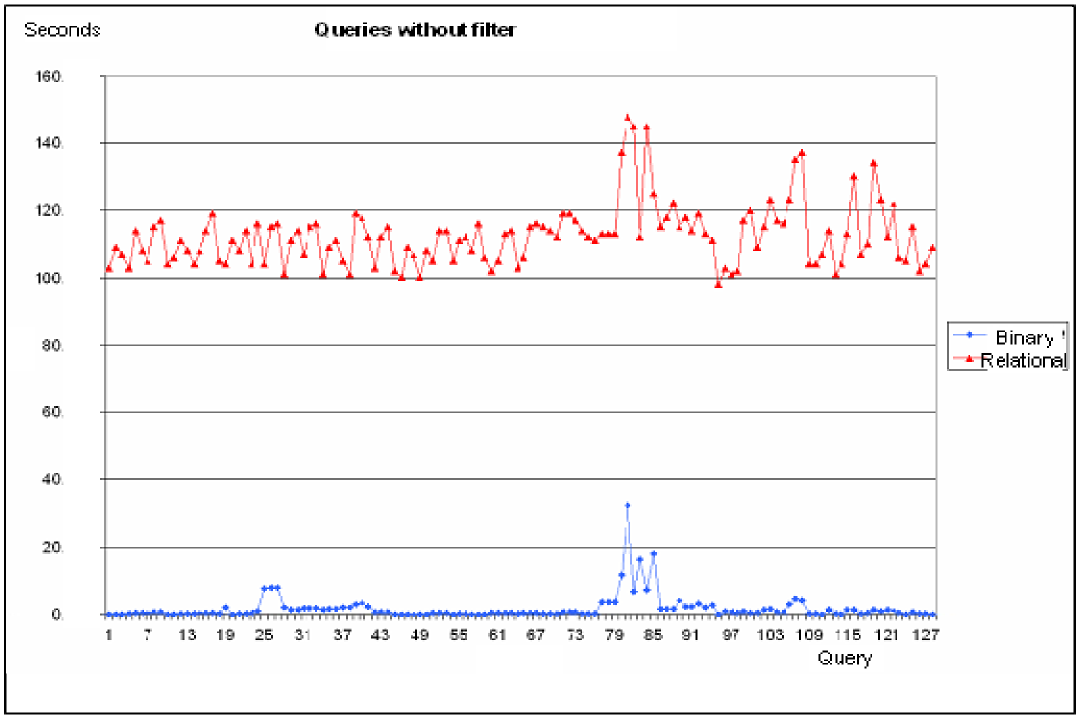


Figure 59 Government Agency's Query Set Run Without Filter

As can be observed on both graphs of this industrial case, the worst query in ADW is much better than the best query on the Traditional DW configuration; this behaviour has been observed as well in the laboratory results (refer to section 6.4.5.1).
The queries were run in both architectures without any modification as they were written in ANSI SQL.

Government Agency's Insert, Update and Delete Operations

This agency was also interested in benchmarking the Insert, Update and Delete operations within the ADW configuration. The results of such measurements are presented in Table 54.

Table 54 Insert Update and Delete Operations

Task	n-ary-Relational (HH:MM:SS)	Binary-Relational (HH:MM:SS)	Savings (%)
Deletion of a Geographical Entity (+ 8 million rows deleted)	00:31:06	00:02:11	92.97%
Reload of a Geographical Entity (+8 million rows re-loaded)	00:43:53	00:21:32	53.07%
Addition of a new column and Update of such column (+103 million rows updated)	>07:00:00 Error 1	00:01:36	
Elimination of a column	Error 2	<00:00:01	

From Table 54 it can be appreciated that ADW had better performance than the Traditional DW configuration on bulk Delete and Insert operations, but in two of the operations Error messages were obtained when the operations were made on the n-ary-Relational DBMS.
The first Error appeared after 7 hours of processing, and it was caused by the lack of space on the *Log* area of the n-ary-Relational instantiation; while the Binary-Relational model instantiation successfully added the new column in only 1.5 minutes. This is because of the way the models work; in the case of the Binary-Relational Model, the addition of another column is quite simple as each column is a Binary-Relation; it does not represent any challenge to add a new column. By contrast, n-ary-Relational DBMSs in general need to recreate the whole table structure and then copy all the information

from the old table to the new extended table structure; for this reason it uses the Database Log area in order to process this request.
Error number 2 was displayed by the n-ary-Relational DBMS because as it was using the compression feature [Poss, 2003] and in order to eliminate a column, it is necessary to uncompress the database and then proceed to the column elimination. This is an important drawback of following the data compression approach suggested by this n-ary-Relational vendor. The Binary-Relational DBMS has not had any problem to eliminate a column.

Government Agency's Cube Construction

This agency uses different exploitation tools, which include Microsoft Excel, Hyperion suite 8.5 and Microsoft Analysis Services 2000; the last one is a tool which builds cube structures to provide OLAP facilities. Measurements of the cube construction were made and the results are given in Table 55.

Table 55 Government Agency's Cube Build

Task	Traditional DW (HH:MM:SS)	ADW (HH:MM:SS)
Cube Build on Microsoft Analysis Services	7,000,000 rows in 19:00:00	+103 million rows in 04:52:00

ADW reduces the time required to build cubes used. The Binary-Relational instantiation built the cube in nearly 5 hours; in contrast the n-ary-Relational instantiation was stopped after 19 hours of processing and it had processed only 7 million rows from a total of 103 million rows. If a linear extrapolation is made, the n-ary-Relational DBMS will require around 280 hours (11.6 days) to complete the cube construction.
An existing problem on this agency's production environment is that tables have been built for each geographical entity in order to construct the cube; in the case of ADW, all rows were stored on a single table and it built the cube in few hours. This is another contribution achieved with the proposed ADW reference architectural configuration: simplifying the Database administration tasks owing to the need of having fewer tables

to be maintained, which was one of the initial motivations of this research mentioned in section 1.4.1.

Government Agency’s Database Administration Tasks

Other aspects measured in this agency were related to the Database administration tasks; the results are shown in Table 56.

Table 56 Database Administration Tasks

Task	n-ary-Relational	Binary-Relational	Savings
Database Backup time (min)	18	10	45%
Indexes building time (min)	120	Not Required	100%
Statistics Computation (min)	960	Not Required	100%
Database Compression (min)	336	Not Required	100%
TOTAL (minutes)	1,434	10	99.3%
(hours)	23.9		

ADW requires fewer database administration tasks, and, as was mentioned in section 7.3.6.1*,* no additional indexes are required, nor statistics computation. In this case, the n-ary-Relational DBMS offers a compression feature as an enhancement for Data Warehouse environments, but this compression feature requires loading data (which requires the total Database space) and then running the compression, which takes 5.5 hours. Hence the reduction on the Database space is achieved. However strictly speaking, the disk space size is not reduced as the space must remain available for certain operations, like the column elimination which requires uncompressing the database.

The total maintenance window has been reduced from nearly one day to ten minutes required by ADW.

Government Agency’s Backup Size

In the previous section, the time to backup the Data Warehouse was measured. However the size of the backup is also important, as it is one of the metrics evaluated. The results for this test are shown in the following table.

Table 57 Government Agency's Backup Size

Task	Traditional DW	ADW	Savings
Backup	64 GB	13 GB	79.69%

The result achieved by ADW in this industrial test is concordant with the 80% savings on Backup size reported in section 7.3.5 when analyzing the results of the laboratory experiments.

Government Agency's Conclusions

ADW has been tested in a Government Agency with good results. Government institutions usually manage large amounts of data. The decisions made by these institutions are relevant for the national development plans. Hence it is important for government to have efficient architectural configurations on their Decision Support Systems and Data Warehouses, and here is where ADW can be relevant for different countries. In Table 58 the main benefits that ADW has provided to the government agency are presented.

Table 58 ADW Benefits for a Government Agency

87% savings in Disk storage costs
55% savings in Load time
+5,000% faster queries response times
The applications did not require any modification in order to use ADW

8.5.3 ADW in a Retail Industry Company

This is a leading retail company with more than 130 stores all over the Mexican territory. The main problem of this company was the long queries response times plus the lack of disk space for growth.

The Environment

This retail company is using Traditional Data Warehouse Configuration, based on an n-ary-Relational instantiation (Microsoft SQL Server). It was compared against an ADW configuration using the Binary-Relational instantiation of Sybase IQ.

Both Data Warehouse configurations were made on the same machine, but during each test the other DBMS was turned off so they did not interfere with each other.
The Decision Support Data Warehouse consists of 25 tables which include Inventory, Departments, Suppliers, Items, and summary tables which contain some pre-calculated results.

ETL processes

In this case, flat files were generated and loaded into each Data Warehouse configuration using the bulk loading utilities of each DBMS, following the same approach as was done in the laboratory. The results of the ETL processes are presented in Table 59.

As can be seen on Table 59, the ADW required 23 minutes to load. It contrasts with the 4 hours and 35 minutes required by the Traditional Data Warehouse configuration. In approximate terms ADW required 8% of the time required to load the same data. This company achieved 92% Load time savings that will impact in their nightly processing window. These results are aligned with the results obtained in the laboratory (section 7.3.2).

It is important to notice that one of the summary tables (Datamart_12_Item_by_unit) took 3 minutes and 45 seconds to load into ADW, but it never finished on the Traditional Data Warehouse configuration; its loading process was cancelled after running for 24 hours. It was removed from the comparison tables in order to compare it with the same set of information.

Table 59 Retail Industry Company ETL Times

Table	File Size (MB)	Rows	Traditional DW (HH:mm:ss:mmm)	ADW (HH:mm:ss:mmm)
Criteria	0.0009	148	0:00:01.000	0:00:00.062
Inventory	1.039	20,765	0:00:01.000	0:00:00.297
Business_Unit	0.010	148	0:00:01.000	0:00:00.094
Unit_Dep_Div_Sum_Table_01	3.260	48,043	0:00:03.000	0:00:01.620
Unit_Dep_Div_Sum_Table_02	3.279	48,275	0:00:03.000	0:00:01.390
Unit_Dep_Div_Sum_Table_03	3.338	48,935	0:00:03.000	0:00:01.453
Unit_Dep_Div_Sum_Table_04	2.022	29,616	0:00:02.000	0:00:00.985
Unit_Dep_Div_Sum_Table_05	3.269	47,626	0:00:03.000	0:00:01.437
Unit_Dep_Div_Sum_Table_06	3.335	48,680	0:00:03.000	0:00:01.484
Unit_Dep_Div_Sum_Table_07	4.173	60,624	0:00:04.000	0:00:01.750
Unit_Dep_Div_Sum_Table_08	4.171	60,835	0:00:04.000	0:00:01.719
Unit_Dep_Div_Sum_Table_09	4.156	60,751	0:00:04.000	0:00:01.687
Unit_Dep_Div_Sum_Table_10	4.230	61,712	0:00:04.000	0:00:01.781
Unit_Dep_Div_Sum_Table_11	4.352	63,560	0:00:04.000	0:00:01.828
Unit_Dep_Div_Sum_Table_12	4.598	67,143	0:00:04.000	0:00:01.953
Item_age	2,546.776	12,469,665	1:09:46.000	0:05:24.562
Item	97.997	419,435	0:01:18.000	0:00:44.188
Department	0.003	69	0:00:01.000	0:00:00.062
Datamart_01_Item_by_unit	700.946	6,117,850	0:26:35.000	0:01:20.094
Datamart_02_Item_by_unit	657.178	5,740,053	0:24:39.000	0:01:29.625
Datamart_03_Item_by_unit	1,137.404	9,907,342	0:46:46.000	0:02:35.766
Datamart_11_Item_by_unit	1,132.418	9,864,686	0:48:11.000	0:02:34.281
Inventory_by_item_unit_last_day	2,675.606	32,320,440	0:57:25.000	0:08:48.437
Supplier	0.096	8,485	0:00:01.000	0:00:00.281
TOTALS	8,993.663	77,514,886	**4:35:16:0**	**0:23:15.836**
		Minutes	**275**	**23**

Disk Space

The next phase of the process is to measure the Database space required for each architectural configuration. The input raw data is 8,993 MB, and again here the presumption that the data will grow once loaded into the Database has been challenged, as the Binary-Relational DBMS decreases the data size.

Table 60 Retail Company's Disk Space

Traditional DW	23,289 MB
ADW	6,787 MB
Disk space savings	70.85 %

Again the Binary-Relational model instantiation used within the ADW configuration, achieved disk space savings of over 70%, as has been previously demonstrated within other industries and with the synthetic TPC-H data set.

Query Response time

In this company, they have their own analytical reports written as Database stored procedures using Microsoft SQL Server proprietary language. Once they were adjusted to the Sybase IQ Database stored procedures language, they ran successfully. The SQL statements within the stored procedures were not changed. The main changes were made because of specific language requirements of each DBMS, as there is not a standard language to write stored procedures.

As this was an initial test, only two analytical reports were ported to ADW. The reports were chosen by the retail company personnel as the more representative analysis reports within their environment. The results of these client reports are presented in Table 61.

Table 61 Retail Company Analyses Reports

Query	Traditional DW (Seconds)	ADW (Seconds)	Faster (%)
Stock_rotation	0.934	0.485	193 %
Top_50_items_on_stock	45.000	4.984	903 %
Totals	**45.934**	**5.469**	**840 %**

From these results the queries against the Binary-Relational instantiation used by ADW, ran 840% faster than the queries in the n-ary-Relational instantiation used in a Traditional Data Warehouse configuration. ADW achieved savings of 89% in query times when compared against a traditional Data Warehouse configuration.

Conclusions

Again the ADW reference architectural configuration has demonstrated its benefits for a Retail industry company. Benefits achieved for the retail company are summarized in Table 62.

Table 62 Benefits for a Retail Industry Company

70% savings in Disk storage costs
92% savings in Load time
+800% faster queries response times
The applications were adjusted to run with Sybase IQ Stored procedures language, as they have been originally written Microsoft SQL Server Stored Procedures language.

8.5.4 ADW in a Telecommunications Industry Company

This is a major mobile phone company in Latin America with more than 100 million clients within the region.

The Environment

In this case, as the volumes were very high and the computing power was large, this company wanted to test on the same production machine to have realistic numbers, as it was hard to have another machine with the same hardware characteristics for testing; The way to conduct the test without disrupting the normal operation was to do all testing in the afternoons, because during the night, the production Data Warehouse is loaded, and then during the morning the users run the query analyses. As during the afternoons the Data Warehouse workload is considerably reduced, it was the testing window time. This pattern of use of the Data warehouse is common according to the observations made during these industrial tests, and represents a good chance to test new architectural configurations in the companies.

The characteristics of the machine used are presented in Table 63.

Table 63 Telecommunications Company's Hardware

	Production Environment	Test Environment
DBMS	Oracle 10G	Sybase IQ 12.5
Operating System	Sun Solaris 5.9	Sun Solaris 5.9
CPUs	6 CPUs	6 CPUs
RAM	12 GB	12 GB

ETL processes

As the Extraction and Transformation times are constants (Section 7.3.1) only the Loading times are presented.

Table 64 Telecommunications Company's Load Times

Table	Number of Rows	Traditional DW (HH:mm:ss)	ADW (HH:mm:ss)
Access_Number	29,088,526	0:40:53	0:51:40
Payment	337,303	0:00:24	0:00:19
Transaction_adjustment	337,303	0:00:05	0:00:02
Call_Detail	18,282,803	0:39:00	0:24:17
Access_fee_and_Charge	2,142,638	0:01:58	0:00:49
Fact_CDE	134,292,483	1:27:00	1:02:34
AHE	3,015,184	0:06:29	0:06:16
TOTALS	**187,183,140**	**2:55:49**	**2:23:49**

From these results, ADW Loaded the same data set faster than the Traditional Data Warehouse configuration, but in this case the difference is not as high as it has been in other cases. This was because of the technology behind the Oracle's Loader utility that has been tuned by the DBA to undertake parallel loads. In this case ADW saved 18% of the loading time.

Disk Space

This company provided information of one day of transactions, reflected mainly in the Fact Table (Fact_CDE), the space results are presented in Table 65.

Table 65 Telecommunications Company's Disk Space

Table	Number of Rows	Traditional DW (MB)	ADW (MB)
Access_Number	29,088,526	8,680.12	5,366.78
Payment	337,303	18.70	39.65
Transaction_adjustment	337,303	2.91	15.03
Call_Detail	18,282,803	5,965.42	3,748.95
Access_fee_and_Charge	2,142,638	236.50	97.81
Fact_CDE	134,292,483	23,690.93	14,473.11
AHE	3,015,184	123.13	188.23
TOTALS	**187,183,140**	**38,117.71**	**23,929.56**

Here again ADW has shown their benefits in space reduction; from Table 65 it can be seen that it achieved savings of 37% in respect to the Traditional Data Warehouse configuration.

Query Response time

For this Telecommunications Company, the query names have been changed to generic names, but the times are real times for these queries.

Table 66 Telecommunications Company's Query Times

Query	Traditional DW (mm:ss)	ADW (mm:ss)
Telco_query_1	8:48	1:32
Telco_query_2	8:48	0:30
Telco_query_3	8:45	0:05
Telco_query_4	8:48	4:48
Telco_query_5	2:22	0:37

From these results, ADW achieved savings of 74% when considering the query set execution and compared against a Traditional DW configuration.

CONCLUSIONS:

For this Telecommunications Company, the Loading times and the space reductions were not as high as they were in the laboratory or for other industries, but the Query response times are extremely fast, running 15,000% faster than the Traditional Data Warehouse configuration which uses an n-ary-Relational instantiation. Benefits achieved for the Telecommunications Company are summarized in Table 67.

Table 67 Benefits for a Telecommunications Company

CPU costs remains the same as both tests were run in the same hardware
Memory costs remains the same as they were run in the same hardware
37% savings in Disk storage costs
18% savings in Load time
15,000% faster queries response time
The applications did not require major adjustments, as they were direct SQL statements.

8.5.5 ADW in the Finance Industry Company 1

This is a financial institution oriented to consumer credits. It has more than 130 business units spread all over the Mexican territory. They generate nearly 3,000 new credits on a daily basis.

The Environment

This financial institution was using the traditional n-ary-Relational database from Sybase, called “Sybase Adaptive Server Enterprise”. As they were having performance problems, the suggestion to use the same vendor Binary-Relational DBMS “Sybase IQ” was made. After seeing the results, the company changed its Data Warehouse to the Binary-Relational model based DBMS, and its production environment is running on Sybase IQ. The characteristics of the machine used are presented in Table 68.

Table 68 Finance Company 1's Hardware Configuration

	Old Production Environment (n-ary-Relational)	New Production Environment (Binary-Relational)
DBMS	Sybase ASE 12.5.3	Sybase IQ 12.6
Operating System	HP-UX 11.23 64 bits	Linux Red Hat 3 32 bits
CPUs	4 CPUs Itanium 64 bits	2 CPUs
RAM	12 GB	4 GB

The main complain they had when using the traditional n-ary-Relational DBMS was a slow query response time. They need to create a lot of summary tables trying to remedy the slow query performance with the implications of more night processing time, which required additional disk storage and increased the database administration complexity. However the most important thing from the business point of view was the difficulty in undertaking complex business analyses because of the slow system performance.

Before considering changing to the Binary-Relational DBMS, they were considering other options:

1. Increase the hardware capacities
2. Change analysis tools (they were using Microsoft Analysis Services)
3. Change to another RDBMS

However, after learning the benefits of the Binary-Relational model, they were willing to do a pilot test. As the test was successful they kept all its Data Warehouse application without disruption. They just changed to the Binary-Relational based DBMS. They were instructed that instead of increasing the hardware, they can reduce the hardware capacities, a decision based on the laboratory experiments conducted on the Heriot-Watt machines and with the other industrial experiences.

Also, once they understood the Binary-Relational Model they understood that a change from one n-ary-Relational DBMS to another brand of an n-ary-Relational DBMS would not make a dramatic change, and the benefits to the company would be marginal. With the experience in this and other companies, people feel satisfied when Database vendors offer improvements of 50% on query response times and according to Stonebraker [2007], they can achieve this just by buying the new generation hardware. People cannot believe that a DBMS based on the Binary-Relational Model can offer hundreds or thousands of times improvements on queries.

Disk Space

In this company the disk space was not the main concern. They were more concerned with the query response time. But as even the space savings are not marginal, it should be highlighted. This financial institution achieved nearly 40% savings in disk space, as shown in Table 69.

Table 69 Finance Company 1's Disk Space

n-ary Relational	13,539 MB
Binary-Relational	8,261 MB
Disk space savings	38.98 %

Query Response time

This was the main reason for this financial institution looking for options in order to offer better response times to the business users that were unable to run complex analyses owing to the long queries response times. Some of the more representative queries are presented in Table 70.

Table 70 Finance Company1's Query times

Query	n-ary-Relational (ss.mmm)	Binary Relational (ss.mmm)	Faster (%)
FPD_Query	0.183	0.094	195%
Contracts	7.526	0.031	24,277%
ICV_Query	10.240	0.047	21,787%
Income	93.930	0.031	303,000%
Ageing	20.005	0.172	11,631%
Management_Level	149.003	0.203	73,400%
Prospects	674.043	0.234	288,053%
Frozen_capital	53.000	0.235	22,553%
Totals	**1,009.109**	**1.047**	**96,381%**

From Table 70, it can be seen that total time for queries was 96,381% faster in the Binary-Relational Instantiation than in the n-ary-Relational instantiation. It is remarkable that the total time has been reduced from 1,009 seconds to just 1 second in the Binary-Relational Model instantiation using less hardware power. In other words, it achieved savings of 99.9%, considering the query execution time.

CONCLUSIONS:

The case of this financial institution is relevant because it was having performance problems mainly on query response time, which was a business stopper because the business analysts were unable to run the proper analyses. Therefore, the IT department was considering increasing hardware or changing from one n-ary-Relational DBMS (Sybase ASE) to other n-ary-Relational DBMS (Oracle or Microsoft SQL Server); but after they understood the benefits of a Binary-Relational DBMS (Sybase IQ), they decided to use it and solve their performance problems, not only gaining in query response times but also in loading times and disk space savings. It is remarkable that this financial institution found its solution with the same vendor (Sybase) when they were looking to change to another Database vendor.

In this case - in terms of computer hardware - instead of increasing it, it was reduced to one third of the computer power in terms of memory, and half in terms of CPUs, even then considerable performance improvements were achieved. With less computer power, the Binary-Relational model instantiation has demonstrated its benefits. In terms of hardware, it will represent computer cost savings of 66% in CPUs and memory, plus 39% savings in disk space.

From the business point of view, the business users are able to run complex analyses and getting the answers within the right times. In the past, they were limited with the n-ary-Relational instantiation. Now, with the reduction from 1009 seconds to just 1 second, the business users are able run the analyses they want.

Benefits achieved for this financial company are summarized in Table 71.

Table 71 Benefits for Finance Company 1

50% savings in CPUs costs
66% savings in RAM Memory costs
39% savings in Disk storage costs
+96,000% faster queries response times
The applications did not require major modifications in order to use the Binary-Relational model instantiation.

8.5.6 ADW in the Finance Industry Company 2

This company manages credit card transactions. An ADW Configuration is now in use to analyze the behaviour of the consumer's transactions, allow the business analysts to obtain valuable information from this data and make business decisions with the facts found in their data. Here again this company was considering implementing another n-ary-Relational DBMS (DB2 or Oracle) to solve their performance problems, but after an initial benchmark between the 3 DBMS, they chose the Binary-Relational based DBMS (Sybase IQ). Today this company is in production with the Binary-Relational based DBMS. Because of the confidentiality necessary to keep with this company, only the main improvements are mentioned, but from here business benefits have been achieved, these are presented in Table 71.

Table 72 Finance Company2's comparison

	Before (Traditional DW conf.)	Today (ADW Configuration)
Average Query Response time	3 to 5 minutes	10 to 15 seconds
Historical Data on line	6 months	18 months
Disk Space	100 GB	120GB
Full DWH Load	6 hours	45 minutes
Data Consolidation Process	30 minutes	5 minutes
Number of rows	450 millions	1,350 millions
Data changes application process	30 hours	30 minutes
Maintenance Tasks	30 hours	3 hours
Online users	45	75
Service Levels	75%	97%

Analyzing Table 72, the following conclusions can be made:
Total query time was 2,233% faster with an ADW configuration than with the Traditional Data Warehouse configuration; in other words, it achieved savings around 70%. The Load time is 800% faster when using ADW.
Regarding other batch processes, the improvements are as follows: Data Consolidation Process (summaries computation) is 600% faster, and Data update "Data Changes application process" was improved dramatically as it ran 6,000% faster; other maintenance tasks ran 1,000% faster.
Considering the business aspects for this company, the number of users was increased by 66% (from 45 to 75). With the space reduction achieved by ADW (60%), this company is able to keep three times as much historical data online (they went from 6 to 18 months and from 450 million rows to 1,350 million row). Even more important is the possibility to enable business users to run analyses against a richer data set, thus obtaining more meaningful answers.
The IT department was able to increase the Service Level (system availability) to the users from 75% to 97% of the time.

CONCLUSIONS:
This financial company has also received the benefits of the Binary-Relational Model incorporated within ADW, and implementing its Decision Support application on top of a Binary-Relational DBMS. The benefits achieved for this financial company are summarized in Table 73.

Table 73 Benefits for Finance Company 2

60% savings in Disk storage costs
800% faster Load time
+2,000% faster queries response times
1,000% faster Maintenance tasks
66% more users are using the system
200% more historical online data
22% more system availability

Since the adoption of ADW, this company has achieved impressive performance improvements in many aspects for their Decision Support System. But, most important, the improvements have enabled business users to obtain richer business analyses when they have more historical data. In addition, they obtained their data analyses results in a faster time to make the appropriate decisions. This company is in production with a Binary-Relational DBMS (Sybase IQ).

8.6 Chapter Conclusions

ADW has been tested in industry, demonstrating its feasibility to perform well in real environments for different industries (Manufacturing, Government, Telecommunications, Retail and Finance). These industrial results are confirmation of the results obtained in the laboratory environment; and organizations can benefit from the use of ADW and solve many of the problems they are actually facing.

Research objectives 4, 5 and 6 have been satisfied as it has been confirmed that the use of an Alternative Data Model within an Alternative Reference Architectural Configuration for Data Warehousing is feasible and brings tangible benefits for organizations. ADW is what has been mentioned in section 1.4.4.

The initial motivations for this research and the objectives established have been fulfilled with both the laboratory test and the industrial tests. The conclusions of this thesis have been made in the body of this thesis, and in Chapter 9 all these conclusion will be put together.

CHAPTER 9
Conclusions and Future Work

9.1 Review of Thesis

This thesis is laid out in nine chapters. This chapter will state the main achievements of each chapter, as well as the issues identified during the research process.

Chapter 1 Introduction:

In this chapter the motivations for this research were exposed. From these the main motivation was to investigate ways to increase data density and reduce data sparsity within the context of Data Warehousing, as this kind of system is growing at an accelerated rate.

In the search of ways to increase data density and reduce data sparsity, compression methods were studied, but alternative data models which abandon the horizontal approach to store and manage data were also investigated. This last approach was preferred as it approaches to the root cause of the problem (the data model beneath the DBMSs); therefore the following research hypothesis was stated:

"Data Density in Data Warehouse Environments can be Improved by Using Alternative Data Models".

The purpose of the research was to design an Alternative Data Warehouse reference architectural configuration (ADW) based on the best suited alternative data model. Consequently, the following thesis objectives were established:

1. Identify an alternative data model, which increases data density (covered in section 6.4.3.2).
2. Find an alternative data model, which better handles data sparsity than the relational model (covered in section 7.3.3.3).
3. Demonstrate that the selected model better handles data sparsity and allows the improvements of data density in Data Warehouse environments by

benchmarking the alternative data model against the relational model (covered in section 7.3.3.3).

4. Create a novel Alternative Data Warehouse reference architectural configuration (ADW), which uses the best suited alternative data model for Data Warehouse Environments (covered in Chapter 8).
5. ADW should maintain compatibility with the widely deployed relational technology (covered in section 8.3).
6. The use of an alternative data model within the ADW helps to solve the storage, querying and maintenance challenges existing in current Relational Data Warehouses (covered in section 8.5).

Chapter 2 Background:

The major background research into the current approaches to manage Data Warehouses was made. A review of strengths and weaknesses of the Relational model was made; also its wide acceptance in industry has been highlighted.

The vast deployment of relational technology - and the SQL language in particular - is why Relational technology has been extrapolated to be used in Data Warehousing, without questioning if this is the best approach to address the Data Warehousing requirements; and this is precisely what has been challenged when proposing ADW.

A traditional Relational Data Warehouse architecture is described here. This architecture has been improved by ADW (refer to Chapter 8)

Also in chapter 2, a revision of existing approaches to increase Query performance was made. It included the use of summary tables [Microstrategy, 1999], materialized views [Akadia, 2007], approximate queries [Acharya, 1999] [Shanmugasundaram, 1999] and cubes [Pendse, 2001a], but none of these approaches goes towards the root cause of the problem which is the underlying data model – the Relational model - and are only technology remedies on top of the Relational Model. In this thesis, a challenge to these approaches is made by using an alternative data model, in this case the Binary-Relational model, which improves the Query performance problem by changing the way data is stored and accessed by changing the horizontal approach and using a vertical approach to store and manage data.

Current approaches to increase Data Density and reduce Data Sparsity, including compression techniques, were also reviewed, but their use in Databases is questionable. Westmann [2000] recommends extending RDBMS to use it, but others like Chen [2001] do not recommend the general use of compression techniques in RDBMS because of its high CPU usage. In this thesis, the use of the Binary-Relational model has been preferred to increase Data Density and decrease Data Sparsity, as this model itself eliminates duplicated values at column level and eliminates Sparsity (refer to sections 3.4 and 7.3.3.3).

Chapter 3 Alternative Data Models:

In this chapter some non n-ary-Relational data models were investigated. The similarity between all of them is that they abandon the Record structure to store and manipulate data.

The studied models were:

- The Associative/Triple Store Model
- The Binary-Relational Model
- The Transrelational Model

Their characteristics, behaviour and storage structures were investigated and reported in this chapter.

In theory, all of them can be useful to increase Data Density and reduce Data Sparsity as none of them stores duplicated values at the column level, therefore it was necessary to benchmark them and measure their behaviour (refer to Chapter 6 and Chapter 7).

Chapter 4 Benchmarking Standards:

In order to benchmark the alternative Data Models against the Relational model, or more accurately the n-ary-Relational model, some existing benchmarks were investigated.

The benchmarks considered were:

- The OLAP Council's APB-1 benchmark
- Microsoft's 1TB benchmark
- The Drill Down benchmark
- TPC's benchmarks (A,B,C,D,H and R)

The TPC-H benchmark in particular was selected as the benchmark to be used for the experiments of this thesis, as it was considered that this benchmark is widely accepted in both research and in industry, and as it avoids the use of extended technology features, like materialised views or indexes. This benchmark helps to measure the Data Model behaviour and not technology-specific implementations.

TPC-H does not consider important tasks involved in the whole Data Warehouse cycle; therefore extensions to the TPC-H have been made in section 4.6

Chapter 5 Implementation of the Storage Mechanism for the Transrelational Model:

It was not possible to find an available implementation of the Transrelational model, and, as a public domain definition of the Transrelational model is available in [Date, 2004], the decision to implement the essential algorithms described there was made in order to benchmark the Transrelational model within the context of Data Warehouse environments. The results of this implementation and its benchmark results had been reported to the research community in Gonzalez [2006a].

As far as the author of this study knows, this was the first public domain implementation of the storage mechanism for the Transrelational model.

Chapter 6 Experiments and Results:

In this chapter, the methodology, the metrics and environment of the experiments are described.

The actual results of benchmarking the alternative data models against the base model used (the n-ary-Relational) are presented.

From the evaluation of some metrics, the Associative/Triple Store model and the Transrelational model were discarded as they failed to satisfy the main research motivation and the stated research hypothesis of increasing Data Density and reducing Data Sparsity; while in contrast the Binary-Relational model had satisfied the main objective. Further experimentation was made with it, and the complete results were presented in this chapter.

Chapter 7 Data Analysis:

In this chapter, the analysis of data obtained from the experiments was made. Based on the analysis, the Binary-Relational Model is best suited to manage Data Warehouse environments effectively. It increases data density, decreases data sparsity, improves dramatically the Query performance, reduces the maintenance tasks, reduces the disk space requirements and in general the processing times.

Chapter 8 The ADW Reference Architectural Configuration:

The final purpose of this research is to offer an Alternative Data Warehouse architectural configuration to organisations which are struggling to manage their Data Warehouses effectively and solve the problems they are facing.

An Alternative Data Warehouse reference architectural configuration, which has been called ADW, is described here. Such architectural configuraton incorporates the use of Binary-Relational DBMSs and at the same time keeps compatibility with the current Relational - or more precisely - with SQL front-end tools that organisations have and in which they have invested significant amounts of money.

In this chapter, some industrial test examples of the application of ADW have been included. These are based on the work following the research carried out at Heriot-Watt University. These results have demonstrated the feasibility of using ADW in industry, and the benefits stated in the body of this thesis are achieved by the organisations which have been willing to implement or test ADW.

9.2 Contributions to Research

The contributions to research of this thesis are:

Contribution 1: Some of the alternative data models have existed for many years in the computer science body of knowledge, but they have been studied for transactional environments, not for Data Warehouse environments as has been done in this research (Chapter 3, Chapter 6 and Chapter 7).

Contribution 2: Extensions to the TPC-H standard benchmark [TPCH2002] have been made in order to consider the full operation cycle of a production Data Warehouse environment (i.e. Backup and Restore times) plus some additional metrics that must be considered, such as the Data Density of a Data Warehouse (Chapter 4).

Contribution 3: The first public domain implementation of the storage mechanism for the Transrelational^{TM} model has been made during this research (Chapter 5) and has been reported to the research community in Gonzalez [2006a].

Contribution 4: A novel Alternative Data Warehouse reference architectural configuration (ADW) has been developed (Chapter 8). ADW includes the use of Binary-Relational DBMSs, and, according to the results reported in Chapter 6 and Chapter 7, the Binary-Relational model is the alternative data model best suited for Data Warehousing environments.

Contribution 5: An explicit classification of existing cube approaches has been made in section 2.4.4. Many cube approaches exist on the research body of knowledge, and as far as is known, no-one has made a clear classification of these cube approaches in the past.

Contribution 6: Improvements to the Transrelational model algorithms have been made in Chapter 5. In particular, the *Column Condensation Threshold* concept has been introduced to Date's proposed algorithm [Date, 2004] to compute the Field Values Table used by the Transrelational model (refer to section 5.4.1) Also *an alternative algorithm* to compute the Record Reconstruction Table has been proposed (refer to section 5.4.2).

9.3 Conclusions

This thesis has considered the use of alternative data models within the context of Data Warehousing to solve the problem of the exponential growth of the Data Warehouses.

This problem is caused by many factors which were the motivation of this research and were listed in section 1.4.1. From these, two of the most important factors related to the

Data Warehouse growth are the low Data Density and the poor management of Data Sparsity. Those two factors are not clear for the user, as they are related more to the DBMS, in which users have little or no control of how the DBMS manages data.

Since the origin of computer science, the record concept has existed, and even today people think horizontally when using the record concept, and as a consequence systems have been designed following a horizontal approach.

Codd developed the Relational model based on the mathematical concepts of Relations, but commercial implementations have not been 100% aligned with such concepts, as is widely known in the research community.

Mathematical Relations can be of any degree, and as a consequence the Relational model as proposed by Codd has been identified in this thesis as the n-ary-Relational model, and sometimes in the body of the thesis it has been referred to simply as the Relational model.

A particularization of the Relational model, called the binary-Relational model, has existed for many years [Titman, 1974; Chen, 1976; Copeland, 1985], in which all the relations are of the 2nd degree. Research has been carried out in the past but within the context of transactional systems where it has not been good and the n-ary-Relational approach has been preferred. Isolated studies have been done with the Binary-Relational model [Boncz, 2002; Stonebraker, 2005] but in both cases the authors are more interested in the implementations of their corresponding DBMS (MonetDB and C-Store) than in the underlying Binary-Relational model itself, as it has been the focus of this research and its broader use in an Alternative Data Warehouse reference architecture, as has been proposed.

By using the Binary-Relational model, the approach to storing and managing data has been changed to means by which data seem to be managed vertically, and by doing this the assumption of managing records horizontally has been challenged.

It has been demonstrated by benchmarking the Associative/Triple Store model, the Binary-Relational model and the Transrelational model, that even when in theory all of them abandon the record structure, the only one that effectively achieved an increase in the Data Density (rows per disk unit) is the Binary-Relational model.

Based on the benchmark results the following benefits have been achieved:

- Reduction in the time required to Load data into the Data Warehouse
- Increase in the Data Density
- Better management of Data Sparsity
- Reduction of disk space, and consequently:
    - Fewer disks to be acquired.
    - Electricity savings achieved by having fewer disks to be powered.
    - The reduction in energy by reducing the cooling BTUs needed to cool the computer centre by having fewer disks generating heat, and even savings when buying a smaller cooler machine.
    - Savings in computer room floor space by having less disk arrays to be accommodated.
    - Savings in secondary storage (tapes) in the same proportion by having fewer disks to be backed up.
    - By having fewer tapes to store, reductions in the size of the room to keep the historical tapes can be also considered.
    - By having fewer disks, the payment of the corresponding maintenance fee to the hardware vendors is also reduced.
- The query execution time is dramatically improved
- Less CPUs and RAM are required. Therefore organisations can get savings in CPUs and RAM acquisitions.
- Maintenance task reduction; this includes:
    - Small backup size
    - Less backup time
    - Less restore time
    - No extra indexes are required; having an impact in two aspects:
        - fewer structures to be maintained (CPU time); and
        - Less disk space.
    - No need to execute statistics maintenance; it positively impacts the CPU time.

- No need or at least fewer summary tables to be maintained, owing to the query response time. This has positive impact on:
    - Less disk space
    - Less processing time to compute the summary tables or materialised views.
    - Fewer indexes on such summary tables
    - Reduction in the number of tables to be maintained, simplifying the DBA maintenance task.

The measurable benefits of using the Binary-Relational model within the context of Data Warehouse have been mentioned, but when it is considered within an Alternative Data Warehouse architectural configuration, like ADW, other benefits are achieved, as the whole Data Warehouse cycle has positive impacts:

- By changing the ETL approach for an ELT approach, the total time of feeding the Data Warehouse is reduced, as the Transformation is made in bulk instead of doing it row by row.
- The Data Warehouse based on a Binary-Relational DBMS has all the benefits listed before.
- The exploitation phase benefits by having faster queries for different analysis tools, including direct SQL statements, Report Writers, OLAP, CRM and Data Mining tools. In the case of tools which use cubes, the cube construction also benefits as the Binary-Relational Repository provides data faster to the cube builder engine.
- Hence this data model is ideal for *ad hoc* queries which are a common requirement in Data warehousing environments.
- Some other benefits are provided by ADW, but these are less tangible or hard to measure:
    - The IT personnel involved in the operation of the Data Warehouse can improve their quality of life by having fewer problems to operate the whole architecture and giving them more time for their own lives. This positive feedback has been obtained from the personnel of two of the organisations which had implemented ADW.

- By having less power consumption (details have been listed before), ADW is contributing positively to environmental issues, as it has less CO_2 emissions to the atmosphere. This is an aspect that is not in the main focus of computer science research, but it is necessary that modern architectures consider their impact over the environment; therefore it can be said that ADW is a "*green friendly*" architectural configuration.

ADW can operate in real life situations: some examples of organisations which have implemented have been mentioned in this thesis, and the results obtained during the industrial tests correspond to the results obtained in the laboratory tests made at Heriot Watt University.

Other conclusions can be made based on the investigation and experimentation carried out in this thesis:

This thesis differs from other studies in the sense that here a broader investigation has been made by including many alternative data models, while others have taken into consideration only one data model in their studies [TriStarp, 2000; Williams, 2003; Boncz, 2002; Stonebraker, 2005; Date, 2004].

The Associative/Triple Store model was the model instantiation which has the lower data density, thus making it less feasible to be used within Data Warehouse environments. Some recommendations to improve this data model instantiation have been made in Chapter 6 and in Chapter 7 and will be summarised in section 9.4 *Future Work* of this thesis.

The Transrelational model as described by Date in Date [2004] is appealing for Data Warehouse environments, but after analysing the results in Chapter 6 and in Chapter 7, it really does not offer the benefits sought in this study. It may reduce the inefficiency of storing repetitive values at column level that exists in traditional n-ary-Relational implementations, but the expected reduction in Data Warehouse size was not achieved. However, a novel public domain instantiation of the storage mechanism for the Transrelational model described by Date has been produced in this research work. Improvements to the algorithms of the Transrelational model have been identified and implemented in this thesis.

Based on the thesis research results and the current state of the Transrelational model, it is argued that it does not represent the model of choice for future Data Warehouse environments.

Disadvantages found:

Analytical Queries generally use few columns to do the analysis, and in such environments the use of the Binary-Relational model adequate, as has been demonstrated in this thesis. However, if the Query needs to reconstruct the whole row and retrieve all the records of the table, like the result set obtained by the SQL Statement

*Select * from table1*

Then in this situation the performance of the Binary-Relational model can be lower than the performance of the n-ary-Relational model. This can happen if the cube tool needs to retrieve all the rows of the table in order to build the cube. This situation must be considered when designing the cubes within ADW.

People realise the advantages they can obtain from implementing ADW, but it is hard to find a justification of request for more investment from the organisations, as frequently they had already invested lot of resources in their current Data Warehouse Architectures.

9.4 Future Work

There is more work that can be done, as this thesis has touched on different models, technology implementations and architectures; the decision of separate future lines of investigation has been made for each of the alternative data models used during this investigation.

Binary-Relational and ADW:

1. Although ADW is not prescriptive on the use of specific tools for each element considered in the architecture, some examples of tools have been mentioned in the body of the thesis. MonetDB and SybaseIQ have been tested, but other Binary-Relational DBMS are being developed - for example C-Store also called

Vertica, Sands Technology [Sand, 2008]; but certainly Binary-Relational DBMS need to mature and gain more acceptance by the industry.

2. Also, it is necessary to develop a methodology to size Binary-Relational Data Warehouses as, according to the results of this thesis research, it is known that the Data Warehouse size will be reduced, but the savings vary from 32% achieved during the laboratory experiments up to savings of 87% obtained on the industrial tests. Organisations need to have more precise figures of expected savings, and it will help to size the hardware required.
3. Investigate the use of the Binary-Relational model within Transactional Systems, as this aspect has not been considered in this thesis.
4. During the experiments the size of the Customer table has not varied highly between the n-ary-Relational, the Binary-Relational and the Transrelational models; therefore a line of research could be started to investigate the circumstances under which the Binary-Relational model will not achieve size reductions.
5. Design, implement and investigate the performance of a hybrid DBMS which keeps one copy of the Data Warehouse on a horizontal repository, and a second copy on a vertical repository. This work has been outlined in Ramamurthy [2002]. This kind of DBMS can be used for both Transactional and Analytical workloads. It has been foreseen in this thesis that, owing to the disk savings achieved by the Binary-Relational model, it will not be necessary to consider doubling the Database size; but at the same time the Query analyzer module should be sufficiently intelligent to determine which copy of the Database can solve the query faster. Another challenge that has been foreseen is the difficulty of keeping both copies of the Database synchronized during Insert, Update and Delete operations, as both models behave differently during these operations.

Associative Model:

1. Do a pure Associative model of the TPC-H database without forcing it to have the 8 tables, and run the benchmark.

2. As the Associative model does not require foreign keys, investigate the benefits of this to the Data Warehouse environments as the Fact table's key is a composed key of all dimensions and all levels, hence a reduction on the total Data Warehouse can be achieved.

Transrelational model:

1. Further research needs to be done on the Record Reconstruction Table (RRT) structure to minimise its size and thus reduce the final DB size, but at the same time it has been envisaged that this will increase the processing time. Date [Date, 2004] identifies another feature that could be used in Transrelational (which may be called Version 3) which uses *Merged Columns*. However, existing drawbacks of the RRT have been identified in this thesis, and they should be the next problem to be tackled before introducing more complexity to the algorithms for the marginal benefits in terms of the final DB size that might be achieved in Version 3.

References

[Acharya, 1999] Acharya, Swarup *et al*. Aqua: A Fast Decision Support System Using Approximate Query Answers. Proceedings 25 VLDB, 1999. Edinburgh, Scotland. pp. 754-757.

[Agrawal, 1997a] Agrawal, D. *et al*. Efficient View Maintenance at Data Warehouses. ACM-SIGMOD 1997.pp 417.

[Agrawal, 1997b] Agrawal, Rakesh. Gupta, Anish. Sarawagi, Sunita. Modeling multidimensional databases. Proceedings of the 13th International Conference on Data Engineering. Birmingham, UK. April 1997. ISBN 0-8186-78070. pages 232-243.

[Ahuja, 2006] Ahuja, Rav. Introducing DB2 9 part 1: Data Compression on DB2 9. http://www.ibm.com/developerworks/db2/library/techarticle/dm-0605ahuja/index.html. Toronto, Canada 24-May-2006.

[Akadia, 2007] http://www.akadia.com/services/ora_materialized_views.html

[Ash, 1968] Ash, W.L. and Sibley, E.H. TRAMP: an Interpretative Associative Processor with Deductive Capabilities. Proceedings of the ACM 23rd National Conference, Brandon/Systems Press. 1968.

[Balakrishna, 1994] Balakrishna, Iyer. Wilhite, David. Data Compression Support in Databases. Proceedings of the 20th VLDB Conference. Santiago de Chile, Chile 1994.

[Ballinger, 2009] Ballinger, Carrie. Relevance of the TPC-D Benchmark Queries: The Question You Ask Every Day.
http://www.tpc.org/information/other/articles/TPCDart_0197.asp.

[Boncz, 1998] Boncz, Peter *et al*. The Drill Down Benchmark. Proceedings of the 24th VLDB Conference, New York, USA. 1998.

[Boncz, 2002] Boncz, Peter. Monet: A next generation DBMS Kernel for query intensive applications. PhD Thesis. Amsterdam, 2002.

[Beyer, 1999] Beyer, Kevin, *et al*. Bottom-Up computation of Sparse and Iceberg CUBEs. Proceedings ACM-SIGMOD 1999. Philadelphia, USA. Pp.359-370.

[Codd, 1970] Codd E.F. A relational model of data for large data banks. IBM Research Laboratory, San Jose, California. Communications of the ACM. Volume 13, Number 6, June 1970.

[Codd, 1990] Codd, E.F. The Relational Model for database Management Version 2. Addison Wesley Publishing Company. ISBN 0-201-14192-2. 1990. USA.

[Cognos, 2008] Cognos Incorporate. Cognos Impromptu and Impromptu Web Reports. http://www.cognos.com/products/series7/impromptu/index.html. 2008.

[Copeland, 1985] Copeland, George P. Khoshafian, Setrag N. A Decomposition Storage Model. In Proc of the ACM SIGMOD Int. Conf. On Management of Data, pp 268-279, May 1985.

[Corey, 1997] Corey, Michael. Abbey, Michael. Oracle Data Warehousing. Osborne/McGraw-Hill. Spain, 1997. ISBN: 0-07-882242-4

[Cormack, 1985] Cormack, Gordon. Data compression in a database system. Communications of the ACM, 28:12, December 1985. pages 1336-1342.

[Chaudhuri, 1997] Chaudhuri, S. Dayal, U. An Overview of Data Warehousing ond OLAP technology. SIGMOD record, 26(1), September 1997.

[Chen, 1976] Chen, Peter Pin-Shan. The Entity-Relationship Model – toward a unified view of data. Massachusetts Institute of Technology. ACM transactions on Database systems, March 1976. Volume 1 Number

[Chen, 2001] Chen, Z. Query Optimization in Compressed Database Systems. ACM SIGMOD record vol.30. No 2, pp 271-282, June 2001.

[CWI, 2008] Centrum voor Wiskunde en Informatica (CWI) web page. www.cwi.nl last updated 2008.

[Darwen, 1995] Darwen, Hugh and Date, C.J. The Third Manifesto. ACM SIGMOD Record. Volume 24, number 1, March 1995. USA.

[Date, 1987] Date, C.J. Where SQL Falls short, Datamation May 1, 1987. USA. pp.83-86

[Date, 2004] Date, C.J. An introduction to Database Systems. Appendix A. The Transrelational Model, Eighth Edition. Addison Wesley. 2004. USA. ISBN: 0-321-18956-6.

[Date, 2005] Date C.J. Database in Depth, Relational Theory for Practitioners. O'Reilly Media Inc. USA 2005. ISBN 0-596-10012-4.

[Date, 2006] Date, C.J. Date on Database: Writings 2000-2006. Apress, Berkeley California, USA. 2006. ISBN-13 (pbk): 97B-1-59059-746-0. ISBN-10 (pbk): 1-59059-746-X

[Datta, 2004] Datta, Anindya, *et al*. Curio: A Novel Solution for efficient Storage and Indexing in Data Warehouses. Proceedings 25th VLDB Conference, Edinburgh, Scotland, 2004. pp.730-733

[Devlin, 1988] Devlin B.A. & Murphy P.T. An Architecture for a Business and Information System. IBM Systems Journal, Vol 27, No. 1 1988. USA

[Dinter, 1998] Dinter, Barbara, *et al*. The OLAP Market: State of the Art and Research Issues. Proceedings of the DOLAP conference, 1998, Washington DC, USA. Pages 22-27.

[Elkins, 1998] Elkins, Steven. Arbor Essbase OLAP Server 5. DBMS June 1998. http://www.dbmsmag.com/9806d08.html

[Feinberg, 2007] Feinberg, Donald. Beyer, Mark. Gartner's Magic Quadrant for Data Warehouse Database Management Systems, 2007. Gartner RAS Core Research note G00151490. 10 October 2007. R2519 10312008.

[Feldman, 1965] Feldman, J.A. Aspects of Associative Processing. Technical note 1965-13, MIT Lincoln Laboratory, Mass. USA. 1965.

[Feldman, 1969] Feldman, J.A. and Ronver, P.D. An ALGOL-Based Associative Language. Communications of the ACM Vol 12. No 8, 1969.

[Freeman, 1997] Freeman, Eva. Birth of a Terabyte Data Warehouse. Datamation, April 1997. USA. pp. 80-84.

[Gill1, 996] Gill, Harjinder S. Rao, Prakash C. Data Warehousing. La integración de información para la mejor toma de decisiones. Ed. Prentice Hall Hispanoamericana S.A. Printed in México 1996. ISBN 0-7897-0714-4.

[Gilly, 1994] Gilly, Daniel. Unix in a Nutshell System V edition. O'Reilly & Associates, Inc. USA, 1994. ISBN: 1-56592-001-5.

[Goldstein, 2000] Goldstein, J. Ramakrishnan, R. and Sahft, U. Squeezing the most out of relational database systems. In Proceedings of ICDE, page 81, 2000.

[Gonzalez, 2004] Gonzalez-Castro, Victor. MacKinnon, Lachlan. A Survey "Off the Record" –Using Alternative Data Models to Increase Data Density in Data

Warehouse Environments. Proceedings of the 21st British National Conference on Databases BNCOD Volume 2. Edinburgh, Scotland 2004.

[Gonzalez, 2005a] Gonzalez-Castro, V. and MacKinnon, L. (authors). Petratos, P. and Michalopoulos, D. (editors). Using Alternative Data Models in the context of Data Warehousing. 1st International Conference on Computer Science and Information Systems by the Athens Institute of Education and Research ATINER. Athens, Greece, 2005. pp. 83-100. ISBN-960-88672-3-1.

[Gonzalez, 2005b] Gonzalez-Castro, Victor. MacKinnon, Lachlan. Data Density of Alternative Data Models and its Benefits in Data Warehousing Environments. British National Conference on Databases BNCOD 22 Proceedings Volume 2. pp 21-24. Sunderland, England U.K. 2005.

[Gonzalez, 2006a] Gonzalez-Castro, Victor. MacKinnon, Lachlan. Marwick, David. An Experimental Consideration of the Use of the Transrelational Model for Data Warehousing. Proceedings of the 23rd British National Conference on Databases BNCOD. Belfast, Northern Ireland, 2006.

[Gonzalez, 2006b] Gonzalez-Castro, Victor. MacKinnon, Lachlan. Marwick, David (authors). Olivares Ceja, Jesus. Guzman Arenas, Adolfo. (editors). A Performance Evaluation of Vertical and Horizontal Data Models in Data Warehousing. Research in Computing Science Volume 22 November 2006. Special issue: Data Mining and Information Systems. Instituto Politécnico Nacional, Centro de Investigación en Computación, México. pp. 67-78 ISSN: 1870-4069

[Gray, 2004] Gray, Peter. King, Peter. Kerschberg, Larry. The Functional Approach to Data Management. Springer–Verlang Berlin Heildelberg 2004. ISBN: 3-540-00375-4.

[Haisten, 2003] Haisten, Michael. The Real-Time Data Warehouse: The next Stage in Data Warehouse Evolution. 2003.
http://damanconsulting.com/company/articles/dwrealtime.htm

[Haughey, 2006] Haughey, Tom. Transforming a Logical Data Model to a Physical Database Design – an overview. http://www.tdan.com/view-special-features/5389 1-Feb-2003.

[Huffman, 1952] Huffman, D. A Method for the Construction of Minimum Redundancy Codes. Proceedings IRE, 40(9) pages 1098-1101, September 1952.

[IBM, 2008a] IBM corporation. DB/2 web page http://www-306.ibm.com/software/data/db2/ updated 2008.

[IBM, 2008b] IBM corporation. WebSphere Data Stage web page. http://www-306.ibm.com/software/data/integration/datastage/ updated 2008.

[Informatica, 2007] Informatica Corporation. Informatica PowerCenter 8.5 brochure. Printed in USA 2007. 6659 (11/06/2007).

[Inmon, 2002] Inmon W.H. Building the Data warehouse 3rd Edition. John Wiley and Sons Inc. 2002. ISBN: 0-471-08130-2

[Kersten, 2005] Kerten, Martin and Manegold, Stephan. Cracking the Database Store. Proceedings of the 2005 CIDR conference. Asilomar, California 2005.

[Kimball, 1996] Kimball, Ralph. The Data Warehouse Toolkit, Practical Techniques for Building Dimensional Data Warehouses. John Wiley & Sons, Inc. USA 1996. ISBN: 0-471-15337-0.

[Koch, 1996] Koch, George. Loney, Kevin. Oracle the Complete Reference. McGraw-Hill inc. Electronic version by Modern Age books Inc. USA. 1996.

[Koomey, 2006] Koomey, Jonathan, The Economist. Los servidores Hambrientos. Expansion Mexico Magazine. Mexico, April 16, 2007. Num 963: ISSN 0185-2728. pp. 58.

[Li, 1999] Li, Jianzhong, *et al*. Aggregation Algorithms for Very Large Compressed Data Warehouses. Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999. pp 651-686.

[Lelewer, 1987] Lelewer, Debra and Hischberg, Daniel. Data Compression. ACM Surveys. Vol 19. No. 3 September 1987.

[Levien, 1967] Levien, R.E. and Maron, M. A Computer System for Inference Execution and Retrieval. Communications of the ACM Vol10, No. 11. 1967. USA.

[Microsoft, 2008] Microsoft corporation, SQL Server webpage. http://www.microsoft.com/sqlserver/2008/en/us/default.aspx . Updated 2008.

[MonetDB, 2004a] MonetDB. Web site ©1994-2004 by CWI. http://monetdb.cwi.nl

[MonetDB, 2004b] MonetDB. An Architectural Overview of MonetDB. http:monetdb.cwi.nl/TechDocs/Core/monet/index.html. updated October 2004.

[Morri, 2002] Morri, Mark. Teradata Multi-Value compression V2R5.0. Teradata white paper, July 2002.

[Mistry, 2001] Mistry, Hoshi *et al*. Materialized view selection and Maintenance using Multi-query Optimization. Proceedings of the 2001 ACM SIGMOD conference on Management of Data. Santa Barbara, California. USA. 2001. pp 307-318.

[Microstrategy, 1999] Microstrategy Inc. Microstrategy Architect User Guide. 09031099, published in October 1999, USA.

[Netezza, 2008] Netezza Corporation. www.netezza.com. 2008.

[Nicholas, 2006] Nicholas, FX. Oracle Data Integrator Technical Overview. An Oracle White paper. California, USA.

[O'Connel, 2003] O'Connel S.J., Winterbottom N. Performing joins without decompression in a compressed Database system. SIGMOD Record, Vol. 32, No. 1, March 2003.

[Oracle, 2005] Oracle Corporation. Oracle Database Concepts 10g release 2 (10.2). Part Number B14220-02. USA 2005.

http://download.oracle.com/docs/cd/B19306_01/server.102/b14220/logical.htm

[Oracle, 2008] Oracle Corporation. Oracle database website. http://www.oracle.com/database/index.html Updated 2008.

[Pendse, 2001a] Pendse, Nigel. Multidimensional data Structures. The OLAP Report. http://www.olapreport.com/MDStructures.htm updated 19-March-2001.

[Pendse, 2001b] Pendse, Nigel. OLAP Benchmarks. Storing Multidimensional data. The OLAP Report. http://www.olapreport.com/MDStorage.htm updated 30-August-2001.

[Pendse, 2003a] Pendse, Nigel. The OLAP survey 4. www.olapreport.com.

[Pendse, 2003b] Pendse, Nigel. Database Explosion. The OLAP Report. http://www.olapreport.com/DatabaseExplosion.htm updated 1-August-2003.

[Pendse, 2005] Pendse, Nigel. OLAP Benchmarks. The OLAP Report. http://www.olapreport.com/products/benchmarks.htm .updated 19-December-2003.

[Pendse, 2006] Pendse, Nigel. OLAP Architectures. The OLAP Report. http://www.olapreport.com/Architectures.htm last updated June 27, 20006.

[Poss, 2003] Poss, Meikel and Potapov, Dmitry. Data Compression in Oracle. Proceedings of the 29th VLDB conference, Berlin Germany, 2003.

[Raden, 2007] Raden, Neil. BI 2.0 What to Expect. 1-Feb-2007. http://www.intelligententerprise.com/showArticle.jhtml;jsessionid=OS0051CJXZ3XCQSNDLPCKH0CJUNN2JVN?articleID=197002610&pgno=2

[Ramamurthy, 2002] Ramarmuthy, Ravishanker *et al.* A case of Fractured Mirrors. Proceedings of the 28th VLDB conference. Hong Kong, China. 2002.

[Redbrick, 2007] http://www-306.ibm.com/software/data/informix/redbrick/

[Roth, 1988] Roth, Mark. Korth, Henry and Silberschatz, Abraham. Extended Algebra and Calculus for Nested Relational Databases. ACM Transactions on Database Systems, Vol. 13, No. 4, USA, December 1988. Pages 389-417.

[Sands, 2008] Sands Technology. www.sand.com 2008.

[Shanmugasundaram, 1999] Shanmugasundaram, Jayavel *et al.*Compressed Data Cubes for OLAP aggregate Query Approximation on Continuous Dimensions. ACM KDD-99. Pages 223-232

[Sharman, 1988] Sharman G.C.H. and Winterbottom N. The Universal Triple Machine: a Reduced Instruction Set Repository Manager. Proceedings of BNCOD 6, pp 189-214, 1988.

[SHQL, 2005] SHQL. http://backpan.cpan.org/modules/dbperl/scripts/shql/

[Software AG, 1997a] Software AG. Adabas D concepts and Facilities Manual. Software AG Dokumantation Uhlandstraße 12, 64297 Darmstadt, Germany. 1997. Manual Order Number: AEE61-006ALL.

[Software AG, 1997b] Software AG. Adabas D Questions and Answers Manual. Software AG Dokumantation Uhlandstraße 12, 64297 Darmstadt, Germany. 1997. Manual Order Number: AEE61-001ALL.

[Simpson, 1997] Simpson, David. Corral your storage management costs. Datamation, April 1997. USA. pp. 88-93.

[Stonebraker, 2005] Stonebraker, Michael. *et al.* C-Store: A column Oriented DBMS. Proceedings of the 31st VLDB conference, Trondheim, Norway, 2005. pp. 553-564.

[Stonebraker, 2007] Stonebraker, Michael. *et al.* One Size Fits all? – Part 2: Benchmarking Results. 3rd Biennial on Innovative Data Systems Research CIDR, California, USA. January 2007.

[Sybase, 2005a] SybaseIQ web site.
www.sybase.com/products/informationmanagement/sybaseiq

[Sybase, 2005b] Sybase Inc. Migrating from Sybase Adaptive Server Enterprise to SybaseIQ white paper. USA 2005.

[Sybase, 2007] Sybase Inc. Sybase Data Integration Suite 1.1 Installation Manual. Printed in USA June 2007. Document ID: DC-00708-01-0110-02. pp.81-82.

[Sybase, 2008] Sybase Inc. Adaptive Server Enterprise web page. http://www.sybase.com/products/databasemanagement/adaptiveserverenterprise updated 2008.

[Teklitz, 2003] Teklitz, Frank. Sybase Business Intelligence and Data Warehousing Solutions for Sybase IQ. Sybase white Paper. Part number L02369. Dublin, California USA. 2003.

[Teradata, 2007] Teradata corporation. Teradata On Line Magazine. http://www.teradata.com/t/go.aspx/?id=115390. USA, 2007.

[Titman, 1974] Titman, P. An Experimental Database System Using Binary Relations. Database Management, Proceedings of the IFIP-TC-2 Working Conference. Ed. Klimbie and Kofferman, North-Holland, 1974.

[TPC, 2009] Transaction Processing Performance Council website. http://www.tpc.org/default.asp, last updated 2009.

[TPCH, 2002]TPC Benchmark H (Decision Support) Standard Specification Revision 2.1.0. San Jose, California, USA. 2002. www.tpc.org

[TriStarp, 2000] TriStarp web site www.dcs.bbk.ac.uk/~tristarp. Updated November 2000.

[US Patent, 1999] U.S. Patent and Trademark Office: Value-Instance-connectivity Computer-Implemented Database. U.S. Patent No. 6,009,432 (December 28, 1999).

[Westmann, 2000] Westmann, Till. *et al.* The Implementation and Performance of Compressed Databases. ACM SIGMOD Record. Volume 29, number 3. September 2000. pp. 55-67.

[Williams, 2003] Williams, Simon. The Associative Model of Data. 2nd Ed. 2003, Lazy Software Ltd. ISBN: 1-903453-01-1. www.lazysoft.com

[Zukowski, 2005] Zukowski, Marcin. Improving I/O Bandwidth for Data-Intensive Applications. Proceedings BNCOD 22 Volume 2. pp 33-39. Sunderland, England U.K. 2005.